SLEC: A NOVEL SERVERLESS RFID AUTHENTICATION PROTOCOL BASED ON ELLIPTIC CURVE CRYPTOGRAPHY

Rania Baashirah

Under the Supervision of Dr. Abdelshakour Abuzneid

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

April, 2019

SLEC: A NOVEL SERVERLESS RFID AUTHENTICATION PROTOCOL BASED ON ELLIPTIC CURVE CRYPTOGRAPHY

Rania Baashirah

Under the Supervision of Dr. Abdelshakour Abuzneid

Approvals

Committee Members

Name	Signature	Date
Dr. Abdelshakour Abuzneid	Shakour Abuzneid	4/26/2019
Dr. Miad Faezipour	Miad Faezipour	4, 26, 2019
Dr. Omar Abuzaghleh		4/26/18
Dr. Sarosh Patel		April 26, 2019,
Dr. Eman Abdelfattah	Eman Abdelfattah	5/6/2019

Ph.D. Program Coordinator

Dr. Khaled M. Elleithy	KM Elleithy	5/17/19

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood	Ausif Mahmood	4-26-2019

Dean, School of Engineering

Dr. Tarek M. Sobh	T. Sobh	5/17/19

SLEC: A NOVEL SERVERLESS RFID AUTHENTICATION PROTOCOL BASED ON ELLIPTIC CURVE CRYPTOGRAPHY

SLEC: A NOVEL SERVERLESS RFID AUTHENTICATION PROTOCOL BASED ON ELLIPTIC CURVE CRYPTOGRAPHY

ABSTRACT

Radio Frequency Identification (RFID) is one of the leading technologies in the Internet of Things (IoT) to create an efficient and reliable system to securely identify objects in many environments such as business, health, and manufacturing areas. Since the RFID server, reader, and tag communicate via insecure channels, mutual authentication between the reader and the tag is necessary for secure communication. The central database server supports the authentication of the reader and the tag by storing and managing the network data. Recent lightweight RFID authentication protocols have been proposed to satisfy the security features of RFID communication. A serverless RFID system is a new promising solution to alternate the central database for mobile RFID models. In this model, the reader and the tag perform the mutual authentication without the support of the central database server. However, many security challenges arise from implementing the lightweight RFID authentication protocols in the serverless RFID network. We propose a new robust serverless RFID authentication protocol based on the Elliptic Curve Cryptography (ECC) to prevent the

security attacks on the network and maintain the confidentiality and the privacy of the authentication messages and tag information and location. While most of the current protocols assume a secure channel in the setup phase to transmit the communication data, we consider in our protocol an insecure setup phase between the server, reader, and tag to ensure that the data can be renewed from any checkpoint server along with the route of the mobile RFID network. Thus, we implemented the elliptic curve cryptography in the setup phase (renewal phase) to transmit and store the data and the public key of the server to any reader or tag so that the latter can perform the mutual authentication successfully. The proposed model is compared under the classification of the serverless model in term of computation cost and security resistance.

ACKNOWLEDGEMENTS

My thanks are wholly devoted to God, who has helped me all the way to complete this work successfully. It is all his grace, mercy, and blessings that make this work possible.

I am honored that my work has been supervised by Dr. Abdelshakour Abuzneid for his patient guidance, motivation, and advice he has provided throughout my research. I have been fortunate to have a supervisor who cares and believes in my work, and who responded to my questions and concerns so promptly. I also would like to thank all my professors, friends, and colleagues at the University of Bridgeport, who helped me to accomplish this work.

I owe a debt of gratitude to my parents for their love and encouragement to pursue my studies in the first place, and my sisters for their immense support.

I finally dedicate this work to my loving and caring daughters; Yara and Maria for their tolerance about being apart from them all this time. My words cannot express how grateful I am to have them both in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INRTODUCTION

The wireless sensor network (WSN) has expanded recently to employ new technologies in the Internet of Things (IoT). The purpose of this evolution is to create a low-cost, reliable, and secure communication network for current and future applications using radio waves most conveniently. Radio Frequency Identification (RFID) is a technology where the detection of the electromagnetic signals in the wireless sensor network identifies objects or people. Hundreds and thousands of RFID applications have been used to improve business efficiency and productivity in a variety of business operations, including supply chain management, access control limitation, product tracking, merchandise allocation, toll collection, and so on. It is also considered an integral part of daily life where its applications not only are limited to business activities, but also everyday life activities that are integrated into cell phones, household, automobile, etc.

The primary system of RFID includes a receiver (reader), transponder (tag), and back-end database (server) to store and manage data. The RFID tag is a label that is placed into the object to be identified and located among hundreds and thousands of objects. It consists of a small antenna attached to a microchip with a small memory to store the object's identity and data [1]. The RFID reader is a scanner placed in a fixed location to interrogate the tag whenever the tag exists in the scanning environment. The

back-end database server operates as a data processor that manages, controls, and stores the data from the tag and reader. An RFID system is depicted in Figure 1.1 [2]. Since the communication channel between the reader and tag is assumed to be insecure, messages in RFID communication are transmitted in clear, and thus are vulnerable to security attacks such as replay attack, impersonation, traceability, man-in-the-middle, desynchronization, denial of service, cloning, and disclosure attack. A secure RFID system must be able to resist different types of attacks through maintaining system requirements of mutual authentication, confidentiality, integrity, availability, privacy, forward and backward secrecy.

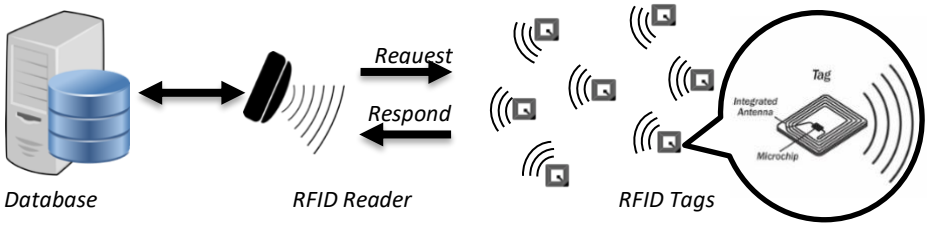

Figure 1.1: Basic radio frequency identification (RFID) model [2]

Since the RFID passive tag has limited resources to compute complex operations [3], the heavyweight protocols are not feasible for practical implementation [4]. On the other hand, lightweight and ultra-lightweight protocols use only simple operations within the tag computation limits and show the lowest tag computation overhead level, so they are mostly used in the current applications. Many RFID protocols are proposed to defend against different attacks. However, several vulnerabilities are detected in the lightweight protocols because it is easy to break out the security of their simple operations.

1.1 Research Problem and Scope

The advanced development of RFID system leads to introducing the concept of serverless RFID where the communication between the RFID reader and tag does not involve a central database. This innovative scheme arises major security issues in the RFID system because both the reader and tags should form an autonomous communication. Multiple serverless RFID protocols are proposed using lightweight operations such as pseudo-random number generator and exclusive-OR operations [5]. Even though these protocols conform to the RFID passive tags limited resources, they are still exposed to security breach due to the lightweight operations used mainly in the reader and tag authentication.

Elliptic curve cryptography (ECC) is a public key scheme for low constrained devices that meet the requirements of the RFID tags. It can provide a security level that is similar to RSA with a smaller key size since their functions are easy to be calculated but hard to be factored back to the original values [6]. ECC is considered in the proposed *SLEC* protocol because it is feasible on the passive tag and provides higher security than other lightweight schemes.

The basic idea behind the *SLEC* protocol is as follows. In the mobile RFID system, a reader and a tag start communicating by authenticating each other without a central database to perform the necessary calculations to establish a secure communication channel. In the authentication session, the reader and tag transmit

challenging messages that can only be computed and verified by a legitimate entity. The transmitted messages should be confidential, and they require encryption and decryption using secure and low-cost operations within the ability of the passive tag to process. The privacy of the tag is also needed to protect the tag secret information and location from being exposed to adversaries. Since the secure algorithms require extensive computations, it is essential to minimize the communication signals in the network, especially when the number of tags is high. We developed a secure and appropriate authentication algorithm that maintain the security of the system and privacy of the tags while minimizing the communication signals in the network to reduce the computation overhead on both the reader and the tag. The proposed protocol is compared with extensive simulations to demonstrate a secure mutual authentication over the currently available protocols. Also, analysis models are developed to validate the proposed solution.

1.2 Motivation behind the Research

Identifying products, humans, or information and authenticate their validity is a crucial matter, especially in mobile RFID systems where the readers and tags exit in a location away from the location of the central database server. In events such as the car dealership industry, a large number of cars needs to be identified and located off the dealership facilities and during the trips between departure and arrival destinations. Identifying asset starts by authenticating the real asset from a fake one along the transportation route. This is done using a secure authentication technique that can be done successfully by a legitimate and registered car. Tracking a car can use the owner's

information to manage transportation securely to avoid stealing cars or losing routes to inventory locations. Authorized facilities must control access to the car's information by allowing only authorized personnel or users to access the car's information. The privacy of the owners is important, so any adversary should not be able to obtain any valuable information to the vehicles or their location. Privacy can be achieved by confusing the adversary by sending noise signals from different locations to avoid capturing the real information or positions of the assets and prevent tracing back the original signal by analyzing the network traffic.

1.3 Potential Contributions of the Proposed Research

Many of the current RFID authentication protocols are proposed to assume a secure setup between the server, the reader, and the tags attached to the cars, which is not realistic in most cases. In our work, we provide a secure setup phase that works as checkpoints through the transportation routes that the cars pass with along their routes to the destination. The setup phase updates the protocol with new values to perform the authentication of the assets. We mislead the adversary about the location of the asset, which will lead subsequently to the location of the source tag. The proposed *SLEC* protocol can handle a group of real and fake signals that are sent from a group of tags at the time to avoid tracing the actual signal. We also secure the asset information using keys that are known only to the reader and tag. The elliptic curve cryptography (ECC) is used in both setup and authentication phases to secure the data forming a noval serverless system. For the fairness of our comparison, we compare the authentication phase of the proposed model with other serverless protocols and ECC-based protocols to validate the

results of our work. Different analysis models are developed to prove the novelty of proposed work.

CHAPTER 2: RELATED WORK

In this chapter, we present the existing authentication protocols in the literature, which mainly rely on the mutual authentication between the reader and tag in the RFID systems. A classification of these protocols and their analysis in term of security, computation, and communication cost is conducted.

Since a passive tag is a tiny chip with scarce resources, it can do only low computations. Hence, RFID protocols are classified in this literature into four categories based on the *complexity of the algorithm* that is used to compute the tag responses: heavyweight, simple weight, lightweight, and ultra-lightweight [7]. Heavyweight algorithms use symmetric and public key cryptography that is beyond the scale of the passive tag ability to process. Simple-weight algorithms use hash functions that are also not feasible for passive tag resources. Lightweight algorithms use simple one-way hash functions, cyclic redundancy checks, and pseudo-random number generators [8]. Finally, ultra-lightweight algorithms use bitwise operations, which can be performed at low cost.

2.1 Recent RFID Authentication Protocols

2.1.1 Heavyweight Protocols

Wang and Sarma [9] proposed two session-based authentication protocols, *SB-A* and *SB-B,* for reader–tag authentication based on symmetric key encryption to ensure

privacy and access control using two types of passive tags. The protocols are based on symmetric cryptography algorithm to provide low-cost authentication such as the Advanced Encryption Standard (AES) and Data Encryption Standard (DES). Protocol *SB-A* in Figure 2.1 includes two processes. The first phase involves mutual authentication between server and tag according to the three-pass mutual authentication protocol according to the International Organization of Standardization and the International Electrotechnical Commission - ISO/IEC 9798-2 [10]. The second phase is for generating a session key between the reader and tag according to the Otway–Rees protocol and updating the pseudo-tag identity (PID). Protocol *SB-B* in Figure 2.2 uses tags with no memory or ID so that all of the tag's information is stored in the server. A physical tag operation is mapped with the virtual digital tag in the server that can do all of the tag's executions. The protocol uses tag nonce and counter control for synchronization, and not the server, because of the limited power of the tag. The protocols proved to be secure against major types of attacks; however, the protocols are considered to be heavyweight, since DES and AES are expensive operations that require a lot of computational overhead.

For traceability issues in RFID, Ryu et al. [11] proposed elliptic curve cryptography-based untraceable authentication protocol (ECU) using the Schnorr signature scheme. The elliptic curve cryptography is considered to be public key cryptography for RFID systems with low constrained tags. It is used to solve the issues of three recent elliptic curve-based untraceable RFID authentication protocols: Strong Privacy-preserving Authentication protocol (SPA) [12], Efficient Mutual Authentication

Server S	Reader R	Tag T
	Step 1: Send RID, OP_R to T →	**Step 2:** Send PID_n and nonce N_T to R ←
	Step 3: Send PID_n, N_T to server ←	
Step 4: Use PID to search the tag K_{TS} **Step 5:** Send E_{KTS} (N_T, N_S, PID_n) to R →	**Step 6:** Send E_{KTS} (N_T, N_S, PID_n) to T →	**Step 7:** - Verify N_T to authenticate S - Send E_{KTS} (N_S, N_T), PID_n to R ←
	Step 8: Send to server ← - E_{KTS} (N_S, N_T), PID_n - RID, OP_R, N_R	
Step 9: - Verify OP_R - Generate K_{RT} - Update PID_n to PID_{n+1} **Step 10:** Send to R - E_{KRS} (N_R, PID_n, RID, OP_R, K_{RT}) → - E_{KTS} (N_T, PID_{n+1}, RID, OP_R, K_{RT}) →	**Step 11:** - Retrieve K_{RT} - Send E_{KTS} (N_T, PID_{n+1}, RID, OP_R, K_{RT}) → - If OP_R is (write), encrypt info with K_{RT} and send it to T →	**Step 12:** - Retrieve K_{RT}, PID_{n+1}, RID, OP_R - Verify OP_R = OP_R in Step1 - Check the on-tag counter - Decode OP_R and execute it - Update PID_n to PID_{n+1} - If OP_R is (read), encrypt info with K_{RT} and send it to reader ←
K_{RS}: server/reader shared key; K_{TS}: server/tag shared key; K_{RT}: reader/tag shared key; N_T: nonce generated by tag; N_R: nonce generated by reader; N_S: nonce generated by server; RID: reader ID; OP_R: operation of reader; PID_n: pseudo-ID of tag in current session; E_K(M): message encrypted by key K.		

Figure 2.1: Session-based authentication protocol (SB-A) by Wang and Sarma

Server S	Reader R	Tag T
	Step 1: Send RID, OP_R to T →	**Step 2:** Send PID_n and nonce N_T to R ←
Step 4: Use PID to search the tag K_{TS}	**Step 3:** Send PID_n, N_T to S ←	
Step 5: - Update PID_n to PID_{n+1} - Send E_{KTS}(N_T, N_S, PID_{n+1}) to R →	**Step 6:** Send E_{KTS}(N_S, N_T, PID_{n+1}) to T →	**Step 7:** - Verify N_T to authenticate S - Send E_{KTS} (N_S, N_T, RID, OP_R), PID_n to R ← - If OP_R is not (kill), update PID_n to PID_{n+1}
Step 9: - Verify reader authorization for OP_R	**Step 8:** - Send E_{KTS} (N_S, N_T, RID, OP_R), PID_n to S ← - Send RID, OP_R, N_R to S ←	
Step 10: - If OP_R = read, send the message - If OP_R = kill: • Send E_{KTS} (N_T, PID_{n+1}, RID) to R → • Kill Vtag	**Step 11:** Send E_{KTS} (N_T, PID_{n+1}, RID) to T →	- Retrieve N_T, PID_{n+1}, RID - Verify RID = RID in step1 - Check on-tag counter with time limit - Perform physical kill operation
K_{RS}: server/reader shared key; K_{TS}: server/tag shared key; K_{RT}: reader/tag shared key; N_T: nonce generated by tag; N_R: nonce generated by reader; N_S: nonce generated by server; RID: reader ID; OP_R: operation of reader; PID_n: pseudo-ID of tag in current session; E_K(M): message encrypted by key K; Vtag: virtual tag in the server.		

Figure 2.2: Session-based authentication protocol (SB-B) by Wang and Sarma

protocol EMA [6], and ECC-based authentication protocol PII [13]. Ryu's protocol generates a digital signature with an appendix on the binary message of arbitrary length, and requires a cryptographic hash function, as shown in Figure 2.3. The sender's session key is combined with the receiver's public key to provide privacy, in which the message can be verified by only the receiver's private key. Ryu's protocol is secure against replay attacks, impersonate attacks, traceability attacks, and it maintains forward security. It requires two scalar multiplications, two hash functions, a message total size of 544 bits, and two communications between tag and reader. Even though this protocol requires complex computations associated with scalar multiplications and a hash function, it does not authenticate the reader.

Server S	**Reader R**	**Tag T**
Setup Phase: - Generate elliptic group G of prime order q. - Choose generator P of group G. - Server private/public keys (y, Y = yP) - Store tag verifier X = xP (public key)		Store x, X, Y (server public key)
Authentication Phase:	**Step 1:** Send random c to T →	**Step 2:** - Pick r as session secret - R = rP - v = H (R, c) - schnorr sign Z = rY, s = r + x * v - Encrypted verifier eid = X ⊕ H (R,s) - Send (eid, Z, s) to R ←
	Step 3: To authenticate tag - Compute R' = y^{-1} Z - Derive X' = eid ⊕ H (R', s) - Check X' = X registered verifier - Compute v' = H (R', c) - Authenticate the tag as H(sP − v' X, c) = v'	
G: Cyclic additive group; P: Generator of group G; q: Order of group G; x_i: Tag's private key; ⊕ XOR; X_i: Tag's public key; y: Server's private; Y: Server's public; H: Hash function.		

Figure 2.3: Elliptic curve cryptography-based untraceable authentication protocol (ECU) by Ryu

To reduce the tag's overhead in heavyweight protocols, Yao et al. [14] introduced The Reviving-UNder-DoS (RUND) authentication protocol to defend against denial of service (DoS) and preserve user privacy by powering up the tag to do complex computing for symmetric and public key cryptography. It leverages the power in DoS scans to enable the tag to respond in two ways: either using simple encryption when low signals from a reader activate the tag, or using public key encryption (higher security) when the backscattered signals are high in an insecure environment. The more signals there are in communication, the more power charges the tag. The option of using public key encryption in RUND protocol is to overcome the problem of breaking up the synchronization state between the reader and tag in symmetric key encryption. The protocol is secure because secret information is not sent in the clear, so no useful information can be gained if any message is compromised. Moreover, the parameters used in communication are changed and updated in every session, as shown in Figure 2.4, to prevent replay attacks, maintain forward security, and resist tracking. Even though the overall efficiency of RUND is O(1), it is still not compliant with the Electronic Product Code Class1 Generation2 (EPC C1 G2) standard [3], which is defined by EPCGlobal Inc. for RFID data communication.

Server: S	**Reader R:** PU_R, PR_R, shared K_i	**Tag T:** PU_R, shared K_i, ID
Initialization Phase:	**Step 1:** Precompute and store in S: $f(K_i, c, pad_1) \leftarrow$ Where pad is padding length for f()	- Counter c is set to 0.
Mutual Authentication Phase:	**Step 2:** Send power waves last for T_{pw} with energy E_c. Send PRN r1 in l length to tag →	**Step3:** Compute: **If Ec energy:** - $I_1 = f(K, c, pad_1)$ - $I_2 = r1 \| f(K, r1 \| I_1, pad_1)$ - $I = I_1 \| I_2$ - Update c = c + 1 - Energy consumed E_{sk} **If E_{pk} energy:** - $E(PU_R, K, r1 \| r2, ID, c)$ in l length - Energy consumed E_{pk} **Step4:** Send I to reader ←
	Step 5: If response with symmetric: - Check counter c and search database for $f(K', c', pad_1) \leftarrow$ - Check r1 for replayed msg. - If matches: tag is authenticated. **If response with public key:** - Check and search database for (ID, K) pair ← - Check r1, r2 for replayed msg. - If matches: tag is authenticated	
Updating Phase:	**Step 6:** Generate r3 and compute $I_3 = r3 \| f(K, r3 \| I_1, pad_1)$ - Send I_3, r3 to tag → - Update $K = f(K, r3, pad_1)$ - Update precomputed $f(K_i, c, pad_1)$ with updated key. - Preserve old key of tag.	**Step 7:** Check I_3 using r3 by computing I'_3 - If matches: reader is authenticated. - Update $K = f(K, r3, pad_1)$ - C = 0
PU_R: Public key of reader; ID: Tag's ID; K_i: Shared symmetric key; c: Counter for current key lifecycle; PR_R: Private key of reader; pad_i: Padding for f(); E_c: The initial power the tag is charged; T_{PW}: Time for the power waves to last; E_{SK}: Energy consumption for hash function; E_{PK}: Energy consumption for public key.		

Figure 2.4: The reviving-under-denial of service authentication protocol (RUND) by Yao

2.1.2 Simple Weight Protocols

To better improve the performance of RFID protocols and reduce the power that is needed for complex operations in ECC-based protocols, Farash [15] proposed a mutual authentication protocol (IECC) based on the elliptic curve. The protocol enhances Chou's authentication protocol (EMA) [6], which does not fulfill the security requirement of

forward security, mutual authentication, tag privacy, and security against location tracking, impersonating attacks, and tag cloning attack for an RFID system. The main idea behind the protocol is to use the server's public key to create the authentication message to avoid breaking the system privacy, as depicted in Figure 2.5. The IECC protocol is secure against major attacks, even though the computation cost is the same as in Chou's protocol that needs to be reduced for practical implementation.

Server S: $\{X_i, yP, P\}$	**Reader R**	**Tag T:** $\{X_i, Y, P\}$
Setup phase: - Generate an elliptic group G of prime order q - Choose generator P of group G - Choose random no. y as a private key - Public key Y = yP - Choose random X from G as tag identifier - Store X_i, Y, P in each tag. **Authentication phase:** **Step 1:** - Choose a prime random no. r - Compute C0 = rP - Send C0 to tags →		**Step 2:** - Choose a prime random no. k - K = kP - C1 = kY - C2 = X_i + h(C0, C1, K) - Send C1, C2 to server →
Step 3: - Obtain K' = y^{-1}C1 - Obtain X_i' = C2 – h(C0, C1, K') - Find a match for X_i' in DB - If found: C3 = h(X_i', K') and tag authenticated - Send C3 to tag →		**Step 4:** - Validate C3 = (X_i, K) - Server is authenticated
G: A additive group of prime order q; P: Generator of group G; h: One-way hash function; y: Server's private; Y: Server's public; X_i: Identifier of ith tag which is a random point in G.		

Figure 2.5: Mutual authentication-based on elliptic curve cryptography (IECC) by Farash

Zhang and Qi [16] also proposed another protocol (EECC) to withstand the security weaknesses of Chou's protocol, EMA [6]. EECC protocol enhances patient

medication safety by also using elliptic curve cryptography. In comparison to the EMA protocol, EECC protocol resulted in better performance and security resistance to impersonate and forward security attacks. However, Baashirah et al. [17] found that Zhang and Qi protocol is vulnerable to forward traceability and reader impersonate attack since an adversary can compromise the private key of the reader by obtaining the tag's secret identifier.

Baashirah et al. improved Zhang and Qi protocol and proposed HBEC protocol that is based on securing the tag's secret identifier using a one-way hash function. HBEC protocol overcomes the security flaws in EECC protocol to provide high security even though the extra hash function adds more overhead to the computation, which should be addressed for the network scalability.

B.Chen [18] proposed a role-based access control (RBAC) protocol for mobile RFID to enable user privacy, role, and access control through the back-end server based on a certification mechanism. RBAC assigns role classes as keys to control the information and the number of times each reader can read a tag. RBAC authorizes readers, assigns role classes to control the reader's authority to request tag information, and updates timestamps using random numbers and different shared keys between the database server and reader and tag ad, as depicted in Figure 2.6. Traceability and replay attacks are prevented using updated random numbers in every session; access control is provided using shared keys to prevent unauthorized readers from requesting or reading any tag's information, and integrity is ensured using timestamps. However, RBAC uses one encryption mechanism that is excessive for low-cost passive tags.

The dispersion spectrum of the conventional single-mode silica fiber has a minimum at 1300 nm region. An increase in the signal attenuation and dispersion will cause a decrease in the fiber length. So at some points in an optical fiber communication link, the optical signal will be regenerated.

Server: kx, ky keys	**Reader:** ky keys	**Tag:** kx keys
1- Reader Authorization and role class:	**Step 1:** Reader sends Hello to tag →	- Create random no. r1 - $M_1 = E_{kx}$ (TID, TS, r1)
	- Create random no. r2. - $M_2 = E_{ky}$ (M_1, r2, RID, Command)	**Step 2:** Sends M_1 to reader ←
- Request role-class command, read tag command, TID, and RID from RBAC - RBAC sends role-class. - $M_3 = E_{ky}$(RID, r1, TS_1, $Cert_R$, role-class) - $M_4 = E_{kx}$(TID, r2, TS_1, role-class) - Send M_3, M_4 to reader →	**Step 3:** Send M_2 to server ←	
	Step 4: - Retrieve r1, TS_1, $Cert_R$, role-class from M_3. - $M_5 = H(TS_1 \oplus r2)$ - Send M_4, M_5 to tag →	
2- Assign No. of reads and update timestamps:		**Step 5:** Verify M_5 using TS_1 from M_4 and its r1 to authenticate reader - Calculate number of reads $TC_{n-1} = TC_n - 1$ - if TS_1 is verified, it's updated to TS_2 - $M_6 = E_{kx}(TS_2, TC_{n-1})$ - Send M_6 to reader ←
	Step 6: - Receive M_6 - $M_7 = E_{ky}(Cert_R, r2, M_6)$ - Send M_7 to database server ←	
Step 7: Retrieve $Cert_R$, r2 from M_7 - If $Cert_R$ is verified, retrieve TS_2, TC_{n-1} from M_6. - $M_8 = E_{ky}(TS_2, TC_{n-1}, r2)$ - Send M_8 to reader →		
	Step 8: - Retrieve TS_2, TC_{n-1}, r2 from M_8 - Verify r2	
TID: Tag ID; K_y: Server/Reader shared key; r: random number; TC_n: number of times a reader request information; K_x: Server/Tag shared key; TS: Timestamp; $Cert_R$: Reader security certificate; RBAC: role-based access control.		

Figure 2.6: Role-based access control protocol (RBAC) by B.Chen

2.1.3 Lightweight Protocols

Successful businesses demand an efficient RFID system that is mainly based on a low computation at a low cost. Many recent RFID protocols use low-cost operations that are handled by low-cost passive tags for practical implementations.

Fernando and Abawajy [19] proposed a mutual authentication protocol for Networked RFID Systems NRS, which is a lightweight mutual authentication scheme for an RFID system using low operations such as exclusive OR operation (XOR) and one-way hash functions. However, Alagheband and Aref [8] reported NRS to be vulnerable to major attacks and specifically a full disclosure attack that compromises the whole RFID system. Alagheband and Aref improved NRS protocol and proposed NRS+ by adding three more hash functions to the authentication message to increase the system security. X. Chen et al. [20] noted that the NRS+ protocol is exposed to desynchronization and traceability attacks by using one random number for the tag and reader. Thus, X. Chen proposed NRS++ to improve the security flaws in the previous versions of NRS by generating two different random numbers, *r1* and *r2*, for the tag and reader using a pseudo-random number generator (PRNG) to defend against replay attack. In Figure 2.7, the authentication message *M3* is encrypted using the tag's random number *r1* and reader's random number *r2* to provide message integrity, so the tag cannot verify any modified message. NRS++ uses fewer hash functions, which resulted in less computation overhead and storage space than the other versions, with more security power.

Server S	Reader R	Tag T
	Step 1: - Generate random no. r - Calculate $M_1 = H(EPC \oplus K_1 \| r)$ $M_2 = r \oplus K_1$ - Send to tag $M_1 \| M_2 \rightarrow$	**Step 2:** - Extract r as $r = M_2 \oplus K_1$ - Compute $C_1 = H(EPC \oplus K_1 \| r)$ If $C_1 = M_1$, generate r_1 $N = r_1 \oplus K_1$ $M_3 = H(EPC \oplus K_1 \| r \| r_1)$ Else termination - $\leftarrow$ Send $M_3 \| N$ to reader
- Update secrets in Database $ID_{new} = ID \oplus (r_{2right} \| K_{1left})$ $K_{1new} = H[(K_{1right} \| r_{1left}) \oplus r_2]$	**Step 3:** - Extract $r_1 = N \oplus K_1$ - Compute $C_2 = H(EPC \oplus K_1 \| r \| r_1)$ - Verify $C_2 = M_3$ If equal: Generate random no. r_2 $M_4 = r_2 \oplus K_1$ $M_5 = H(EPC \oplus K_1 \| r_1 \| r_2)$ If not equal: terminate - Send $M_4 \| M_5 \rightarrow$	**Step 4:** - Extract r_2 as $r_2 = M_4 \oplus K_1$ - Compute $C_3 = H(EPC \oplus K_1 \| r_1 \| r_2)$ - Verify $C_3 = M_5$ If equal: Update the secrets. If not equal: terminate
ID, EPC: Tag identifier; H(): one-way hash function; K_1: Server/Tag shared key; r, r_1, r_2: random No; $\oplus$/$\|$: XOR and concatenation operation.		

Figure 2.7: Mutual authentication protocol for networked RFID systems (NRS++) by X. Chen

C. Chen [21] proposed Anti-Counting Security Protocol (ACSP) as another lightweight protocol for RFID systems to defend from a counter attack, which is defined as the attacker's ability to count the number of objects in a system. Safkhani et al. [22] reported ACSP to be vulnerable to major attacks, including the forward/backward traceability attack. Safkhani further proposed ACSP+ to improve Chen's protocol. Later, X. Chen [20] pointed out that ACSP protocol is not secure and proposed ACSP++ to withstand DoS and forward/backward traceability attacks. ACSP++ enhances the session identifier (SID) update, which is used to verify the current session, and tag identification

phases that suffer from different attacks in ACSP and ACSP+ versions. In ACSP++ as depicted in Figure 2.8, a tag identifier (TID) is added to the identification message as ($\overline{IDENT}$, R4, R5, TID) instead of ($\overline{IDENT}$, R4, R5), and the authentication message (($\overline{AUTHEN}$, R4, R5, TID) is replaced with ($\overline{AUTHEN}$, R5, TID) to overcome DoS attack and to modify the TID in the identification phase. The update phase of every key is associated with two separate nonce values to avoid forward and backward traceability.

Reader R	**Tag T**
(SID Update Phase) Step 1: - Generate nonce R1 - Send the following to tag: $\overline{UPDSID}$, R1⊕SID, H($\overline{UPDSID}$, R1, SID) →	**Step 2:** - Extract R1 to verify H($\overline{UPDSID}$, R1, SID) - Generate R_2 - Update SID SID_{new} = H(SID‖R2‖R1) SID_{old} = SID_{cur}
Step 3: - Extract R2 and verify H($\overline{UPDACK}$, R2, R1, SID) - Update SID as SID_{new} = H(SID‖R2‖R1)	- ← Send to reader confirmation: $\overline{UPDACK}$, R2 ⊕ SID, H($\overline{UPDACK}$, R2, R1, SID)
(Tag Identification Phase) Step1: - Generate R3, R4 - Send the following messages to tag → a) $\overline{SELECT}$, SID1⊕ R3, H($\overline{SELECT}$, R3, SID)) b) $\overline{QUERY}$, SID ⊕ TID ⊕ R4, H($\overline{QUERY}$, R4, SID, TID))	**Step 2:** - Extract R3' to verify H($\overline{SELECT}$, R3, SID) If not verified: wait until next run. If verified: respond with step3.
	Step 3: - Extract R4' to verify H($\overline{QUERY}$, R4, SID, TID) - Generate R5
Step 4: - Authenticate tag - Extract R5' to verify H($\overline{IDENT}$, R4, R5, TID) - If not verified: stop the session and send $\overline{QUERY\ REF}$ →	- ← Send ($\overline{IDENT}$, TID ⊕ R5, H($\overline{IDENT}$, R4, R5, TID)
- If verified: update TID as TID_{new} = H(TID‖R4‖R5) TID_{old} = TID - Send ($\overline{AUTHEN}$, H($\overline{AUTHEN}$, R5, TID) →	**Step 5:** - Calculate and verify H($\overline{AUTHEN}$, R5, TID) - If not verified: stop the session. - If verified: update the tag identifier as TID_{new} = H(TID‖R4‖R5)
R1, R2, R3, R4, R5: nonce; $\overline{SELECT}/\overline{QUERY}$: Select/ query commands; SID_{cur}/ SID_{new}: Current/ New session identifier; $\overline{UPDSID}/\overline{UPDACK}$: SID update/ Update knowledge message; TID_{cur}/TID_{new}: Current/ New unique identifier; $\overline{IDENT}/\overline{AUTHEN}$: Identification/ authentication messages.	

Figure 2.8: Anti-counting security protocol (ACSP++) by X. Chen

Even though the protocol improved the security weaknesses of all of the ACSP versions, it did not lower the computation overhead nor the storage space.

Chien and Huang [23] presented LAP, which is a lightweight authentication protocol to solve the vulnerabilities in the authentication protocol of Li et al. [24] and enhance the computational cost from O(n) to O(1) in identifying tags in RFID systems. The security of LAP protocol is based on a synchronized PRNG between the reader and tag using a secret key, secret ID, and index pseudonym. In Figure 2.9, LAP protocol uses the rotate operator on the message and left/right operator for the divided rotation during the messages that were exchanged to form a secure permutation. Random numbers are used to shift the secret values of the tag to be used safely in communication. Then, the random number is XORed with the shifted secret value to retrieve a tag by the server securely. The server uses the index pseudonym (IDS) to quickly identify the tag in the database instead of computing PIDL $\oplus$ PIDR for every tag to make the computation O(1).

LAP protocol is resistant to replay attack, DoS, and forward security. It can be employed easily by different standards such as EPC Gen2 and ISO 15693 [25] for practical implementation. However, the protocol was noted as being partially secure against traceability and synchronization attacks, since a tag can be traced between two successful sessions if the tag could not update its IDS.

Burmester and Munilla [26] proposed a lightweight mutual authentication protocol called Flyweight that is based on exchanging messages using only PRNG. Their

Server S: flag, X_{old}, X_{new}, IDS_{old}, IDS_{new}, SID	**Reader R**	**Tag T:** {SID, IDS, X}
	Step 1: - Generate R_1. - Send Query$\|\|R_1$ to T →	 - Generate R_2 - Compute g' = g($R_1\|\|R_2\|\|X$) - SID' = rotate(SID, g') - R' = left(SID' ⊕ g')
Step 2: - Search IDS_i - <u>If IDS == IDS_{old}</u>: flag = 0, X = X_{old} - <u>If IDS == IDS_{new}</u>: flag = 1, X = X_{new} - g' = g($R_1\|\|R_2\|\|X$) - SID' = rotate(SID, g') - Verify R' as R' = left(SID' ⊕ g') - Compute R'' = right(SID' ⊕ g') - <u>If flag = 1</u> • IDS_{old} = IDS_{new} • X_{old} = X_{new} - Else • IDS_{new} = g(IDS||SID') • X_{new} = g(X||g') - Send R'' to reader →	- Forward R1$\|\|R_2\|\|$R'$\|\|$IDS to S ←	- Send $R_2\|\|$R'$\|\|$IDS to R ←
	- Forward R'' to T →	**Step 3:** - Verify R'' right(SID' ⊕ g') - Update: • IDS = g(IDS||SID') • X = X_{new} = g(X||g')
Step 4: - When OK is received, send SID to R →	- Forward ACK to S ←	- Send ACK to R ←
SID: Secure ID; PID: Partial ID; IDS: Index pseudonym; g(): Random No. generator; X: l-bit secret key; R_1, R_2: Random numbers; Rotate(): Rotation function; Left(s): Left half of s; Right(s): Right half of s; ACK: Acknowledgement.		

Figure 2.9: Lightweight authentication protocol (LAP) by Chien

protocol is based on a shared PRNG algorithm between the tags and back-end server that takes the same seed to produce the same output. The concept of the protocol is to use three consecutive numbers—RN1, RN2, and RN3—generated by the same PRNG in the server, and the tags of five numbers if an active adversary is presented, such as in Figure 2.10. Furthermore, RFID tags precompute the values to the server challenging the response so that an adversary can be detected based on the response time from the tag. The protocol can provide mutual authentication, integrity, confidentiality, and forward

and backward security. Besides, it provides robust synchronization, since the server keeps a record for the current and next response value of the tag.

Server S	Reader R	Tag T
	Step 1:.	
	- Send Query to T →	- RN1 = g_{tag} (state) - Set alarm cnt = 1 - Send RN1 to R ←
	Step 2:	
- Check if RN1 = $RN1^{cur}$ • cnt = 1 • Generate RN2, send RN2 to R →	- Forward RN1 to S ←	
- If RN1 = $RN1^{next}$ • cnt = 0 • Update values in DB • Send updated RN2 to R →		
		Step 3:
	- Forward RN2 to T →	- If RN2 is correct to authenticate S • Generate RN3, RN4, RN5 • Cnt = 0 - If cnt = 0, send RN3 to R ← - If cnt = 1, send RN4 to R ←
Step 4:		
- If RN = RN3, and cnt = 0 • Tag is authenticated	- Forward RN4 to S ←	
- If RN = RN4 • Send RN3, store RN5 • Update values • Send RN3 to R		
		Step 5:
	- Forward RN3 to T →	- If RN3 is correct and cnt = 1 • Send RN5 to R ← - Else terminate
Step 6:		
- If RN5 is correct • Authenticate T • Update values	- Forward RN5 to S ←	
- Else terminate		
RN: Random numbers output of the same generator function		cnt: l-bit flag

Figure 2.10: Flyweight mutual authentication protocol by Burmeter and Munilla

S. Lee et al. [27] proposed a lightweight protocol (MASS) for RFID systems using XOR and a one-way hash function to conform to the scarce resources of RFID tags. The concept of the MASS protocol is to challenge the tag with a fresh random string

every session, and the tag responds using the reader's value and its own random key to authenticate the reader ad, as depicted in Figure 2.11. The secret key is shared between entities, and all of the messages are encrypted during transmission. However, Zuo [28] conducted a survivability experiment on the authentication protocol proposed by S. Lee et al. and defined the vulnerability of the protocol to replay, desynchronize, and impersonate attacks. Zuo concluded from his experiment that the system could employ two different values for the keys (old, new) to recognize the tag and overcome the desynchronization problem.

Server S	Reader R	Tag T
	Step 1: - Generate l-bit string str - Send str to tag →	**Step 2:** - Generate l-bit string rA - rB = h(rA ⊕ Ki ⊕ str) - rC = h(rB ⊕ Ki ⊕ str) - Send rB, rC to reader ←
	Step 3: - Search database to match key Ki - If found proceed to update key - Retrieve rB from rC - Ki = h(Ki) - r'C = h(rB ⊕ Ki ⊕ str) - Send r'C to tag →	**Step 4:** - Verify r'C = rC - If verified, update key
Ki: Tag/server shared secret key; h(): One-way hash function		

Figure 2.11: Lightweight protocol based on synchronized secret (MASS) by S. Lee

To reduce the communication time during the authentication session, K. Lee et al. [29] proposed Efficient Passively-Untraceable Authentication Protocol (EP-UAP). The concept of EP-UAP is that the system precomputes all of the necessary computations before the system initialization, so only low computation overhead is required on the tag side during the process phase. The protocol is based on Randomized Hash-Lock protocol,

which uses a static identifier, and its strong security against traceability depends mainly on PRNG to randomize the responses, as explained in Figure 2.12. Since precomputing all of the possible random numbers and responses requires a storage memory for all of the precomputed data in the database, EP-UAP is preferred for small to medium networks, as the storage memory increases when the number of tags increases. The protocol shows considerable improvement over the randomized hash lock protocol in terms of computation time, in that only requires 40 ms for authentication; this is similar to LRMAP, which is the most efficient one in stateful protocols. However, it requires 100 MB of database storage memory. The protocol provides integrity due to the two randomly generated nonce values that are used from both tag and reader and is secure against passive attacks and traceability due to the random responses. However, the EP-UAP protocol seems to be vulnerable to active attacks such as impersonate and replay attacks, since the random responses depend on the database/reader. It also requires high storage capacity in the database side.

Reader	Tag
Step 1: - Generate R_R - Send Query, R_R to tag →	
	Step 2: - Generate R_T - Compute $m_{TR} = H(ID_{1T}\|\|R_R)$ - Send m_{RT}, R_T to reader ←
Step 3: - Search for ID^i_{1R} - Verify $H(ID^i_{1R}\|\|R_R) = m_{TR}$ to authenticate the tag. - Compute $m_{RT} = H(ID^i_{2R}\|\|R_T)$ - Send m_{RT} to tag→	
	Pre-compute $c_T = H(ID_{2T}\|\|R_T)$
	Step 4: - If $m_{RT} = c_T$, reader is authenticated.
H: One-way hash function; ID: Tag identifier; R_R, R_T: nonce generated by reader/tag; m: Authentication challenge; c: Authentication challenge response.	

Figure 2.12: Efficient passively-untraceable authentication protocol (EP-UAP) by K. Lee

To defend against a desynchronization attack, Rahman and Ahamad [30] proposed a Desynchronization attack-resistant Robust Authentication Protocol (DRAP) in the wireless identification and sensing platforms (WISP), where RFID technology is combined with sensor nodes. Their protocol mechanism is to decrease the tag collision that leads to DoS attack, as shown in Figure 2.13. The technique is to reduce the collision rate at the link layer and maintain the system's efficiency. The protocol also detects the DoS attack and recovers the synchronization state of the system. It has higher resources than passive tags, which allow higher security implementation. Yet, it has a short distance limitation, where tags can only function less than 1–2 m away from readers.

Server S	**Reader R: ID_i: K_{iprev}, K_i, D_{iprev}**	**Tag T: K_i, ID_i, Δ**
	Step 1: - Generate random n_r. - Send n_r to tag →	
		Step 2: If ($\Delta \leq D_{new} - D_{old}$) - Generate random n_i - $\alpha_i = P(K_i \oplus n_r \| n_i)$ - $\beta_i = E_{Kwti}(h(ID_i) \oplus D_{new})$ - Send α_i, β_i, n_i to reader ←
	Step 3: - Generate $P(K_i \oplus n_r \| n_i)$ for all tags to verify α_i. <u>If there is a match:</u> - Decrypt α_i and β_i - Retrieve D - If D_{newi} is not equal to D_{new} then update: $Ki_{prev} = Ki$ $X = h(Ki)$ $\alpha j = P(X \oplus n_r \| n_i)$ $Ki = h(x)$ $Di_{pev} = Di_{new}$ - Else ignore the message and αj = rand <u>If there is no match:</u> - Generate $P(K_{iprev} \oplus n_r \| n_i)$ for all tags to verify α_i - If correct: - Decrypt α_i and β_i - If D_{newi} is not equal to D_{oldi} then update: $\alpha j = P(h(Ki_{prev}) \oplus n_r \| n_i)$ $Di_{prev} = Di_{new}$ - Else ignore the message and αj = rand Else ignore the message and αj = rand - Send αj to tag →	
		Step 4: - Y = h(Ki) - Generate $P(Y \oplus n_r \| n_i)$ to verify αj - if correct: Ki = h(Y)
P(): Pseudorandom No. generator; Δ: Activity threshold; D: Sensor value; K_i: Secret number; ID: Tag identifier; h(): One-way hash function.		

Figure 2.13: Desynch attack-resistant robust authentication protocol (DRAP) by Rahman

Authentication in most RFID protocols is executed between one reader and one tag at a time. Liu et al. [31] proposed a grouping proofs-based authentication protocol (GUPA) to enable authenticating multiple tags and multiple readers simultaneously, such that multiple readers can authenticate a single tag, and a single reader can authenticate multiple tags in large-scale RFID. GUPA protocol is based on hierarchical identification between independent subgroups in a distributed RFID system, and the use of an asymmetric denial mechanism to resist denial-of-proof attack (DoP). For the anonymous authentication of a new entity, GUPA deploys a ring signature using lightweight cryptography (elliptic curve). It also uses lightweight bitwise operations for readers and tags secret information updates, PRNGs, one-way hash functions, timestamps for session freshness, and access lists for each legal reader/tag during system initialization as identity flags to prevent forgery and tracking attack, as fully explained in Figure 2.14. Since the flags are chosen randomly from the pseudonym index, queries and responses are independent for each session to resist DoP attack; hence, illegal proofs are eliminated during authentication.

Database: DB	**Reader: R_j**	**Tag: T_a**
Initialization Phase: 1- Generate PRN r_{DB} 2- Send r_{DB} to tag → 6- Verify H_1 in database for match 7- $H_1 = (\Delta R_j \| L_R \| r_{Ty})$ 8- PRNG (ΔR_j) 9- Send $H_1 \| PRNG (\Delta R_j)$ to tag → **Authentication Phase:**		3- Generate r_{Ty} 4- $H_1(L_R \| r_{DB})$ 5- Send $r_{Ty} \| H_1(L_R \| r_{DB})$ to DB ← 10- $PRNG^{-1}(\Delta R_j)$ to obtain ΔR_j 11- $H_1 = (\Delta R_j \| L_R \| r_{Ty})$ to authenticate DB 12- Add ΔR_j to L_R
L_R: Local access list; ΔR_j: Reader's information; H(): One-way hash function.		

Figure 2.14: Grouping proofs-based authentication protocol (GUPA) by Liu for a single-reader—single-tag case

Since tag collision is a major problem in the large-scale networks, Rahman and Ahamad [32] proposed two probabilistic batch authentication protocols to determine the valid tags efficiently and accurately in large-scale systems. FTest is a protocol based on Frame Slotted Aloha algorithm that is used to reduce the probability of collision slots. The other protocol is GTest, which is a protocol based on group batch authentication that is used to reduce the cost of detecting counterfeit tags. Their protocols use simple lightweight operations such as XOR and cyclic redundancy checks (CRC) with a shared key for each group of tags. The theory in both protocols is not to send the tag ID when responding, but rather accept or reject a tag by estimating the number of fake tags. In the FTest protocol that is depicted in Figure 2.15, a counterfeit threshold parameter is used in the system to reduce the number of rounds in the detection process and response time of the protocol, so that the entire tag responses do not need to be checked. Instead, the detection will stop if the percentage of counterfeit tags exceeds the counterfeit threshold. In GTest, the reader randomly selects a population of tags to authenticate. If one counterfeit tag is detected, the batch of tags will be considered invalid. The reader needs to read a large amount of data to identify the validity of a batch in GTest, so the reader still consumes time through the computation overhead from the tag search. Both FTest and GTest protocols are proved to be secure against tracking and privacy attacks since tags responses are based on dynamic frame size, random numbers, and ID that is not transmitted during communication. However, the FTest shows less execution time and better performance over GTest.

Server S	Reader R	Tag T: Shared group key k_i
Group Identification Phase:	1- Send nonce n_r to tag→	2- Respond by $h(k_i \| n_r)$ ←
	3- Find a group key to decrypt the message. 4- Identify the group of tags based on the group key.	
Authentication Initialization Phase:	1- Send to server "Start authentication" ← 2- Receive (f, r) from server 3- Broadcast frame size and random no.	4- Each tag compute its slot position SP = h(id, r) mod f = 0 or 1 5- Send SP to reader with random bits ←
	6- Generate RV based on responses 0, 1, coll. 7- Turn collision slot into singleton by removing one tag (removed tags remain silent until next phase) 8- Send RV to server for verification.	2- Respond $h(id \| n_r)$
	1- Send random n_r from server to rem tags →	
Counterfeit Detection Phase:	3- Forward RV to server ←	
4- Reconstruct RV_S as only valid tags can compute correct h() 5- Accept valid tags if RV_S = RV		
n: Nonce value; k_i: Shared group key; h(): One-way hash function; SP: Slot position within frame; id: Tag ID; f: Frame size; r: Random N; RV: Response vector generated by reader; RV_s: Response vector generated by server; rem: Set of tags removed to reduce collision slot.		

Figure 2.15: Batch authentication protocol based on Frame Slotted Aloha (FTest) by Rahman

Another anti-collision security protocol (ACS) is proposed by Keqiang et al. [33] for a high-efficiency RFID system combining the chaotic sequence generator with the dynamic frame-slotted ALOHA algorithm for fast tag identification. The protocol scheme is based on a logistic mapping structure with XOR operation and spreading operation to generate real-time keys in a chaotic sequence that are used in authentication messages. Keys are updated in each response from tag to reader and reader to tag during the same session using iteration equations that are known only to the server and tag, such as in Figure 2.16. The protocol is effective against counterfeits and impersonates attacks, as the authentication scheme not only depends on the iterated key but also on spreading

code and random numbers, so faking at least one of them will result in a wrong response. The protocol requires only four message exchanges, low hardware cost, and low computation cost on the tag side. It also has lower energy consumption than other heavy and simple weight protocols because XOR uses less energy than symmetric encryption and hash functions.

Server S: K_0	**Reader R**	**Tag T: K'_0**
	Step 1: - Generate and send a frame size R_0 to tag →	**Step 2:** - Receive R_0. - Choose slot index with the value in [1, R_0] - Reset time slot counter = slot-index - r = 20, i = (r+R_0), $x'_0 = K'_0$ - $x'_{k+1} = rx'_k (1-x'_k)$ iteration = x'_i - x'_i = ChaosSpec - Perform one-time iteration to get $K'_{i+1} = x'_{i+1}$ - $H'(R_0) = R_0 \oplus K'_{i+1}$ - $H'(ID) = K'_{i+1} \oplus ID$ - Generate random R_1
- K_0 = Master key, $x_0 = K_0$ to compute x_i - Verify ChaosSpec using x_i: • If there is collision, go to step5. • If no collision, proceed. - Perform one-time iteration to get $x_{i+1} = K_{i+1}$ - Extract R'_0 from $H'(R_0)$ and verify $R'_0 = R_0$ - Tag is authenticated. - Extract ID from H'(ID) - Perform R_1 iteration to get x_j, j = (r+R_1+R_0) - $K_j = x_j$ - $H(R_1) = R_1 \oplus K_j$ - Send to reader $(H(R_1)) \otimes$ ChaosSpec →	**Step 3:** - Send $(H'(R_0)\|H'(ID)\|R_1) \otimes$ ChaosSpec, and R_0 to S ←	- Send $(H'(R_0)\|H'(ID)\|R_1) \otimes$ ChaosSpec ←
	- Send $(H(R_1)) \otimes$ ChaosSpec →	**Step 4:** - Perform the equations to get $K'_j = x'_j$ - Calculate $R'_1 = H(R_1) \oplus K'_j$ - If $R'_1 = R_1$, then $K'_j = K_j$ - R is authenticated
	Step 5: Collision case - Increase tag's slot counter by 1 - Restart identification process in Step2 **Step 6: No authentication occurs** - Issue AdjustQuery command - Adjust R_0 to decide a new frame size - Send search signal to rest of tags →	
R_0: Frame size; i: Number of iterations; K'_0: Tag key; K_0: Server master key; K'_{i+1}: Real-time key; H(), H'(): One-way hash functions; ChaosSpec: Spreading code; ID: Tag's ID; R_1: Random number generated by tag; r: Constant value to put the equation in chaotic state.		

Figure 2.16: Anti-collision security protocol (ACS) by Keqiang

Cho et al. [34] proposed a hash-based mutual authentication protocol (HBA) to defend against the brute force attack. This protocol was reported by Chang et al. [35] to be vulnerable to denial of service (DoS) and replay attacks. Later, Chang et al. proposed an improved (HBA+) protocol to avoid DoS and replay attacks using a shared PRNG algorithm between the server and tag to produce the same output that is used in updating the protocol values, as in Figure 2.17. Also, the confidentiality in the protocol is based on protecting the secret value *datai* using reader ID (*Rid*), which is only known to a legitimate reader and server. The improved protocol of Chang is considered to be efficient and secure against DoS attack, traceability, and forward secrecy.

Server S	**Reader R**	**Tag T**
	Step 1: - Generate random No. R_r →	**Step 2:** - Generate random No. R_t - M_1 = Auk ⊕ I ⊕ PRNG (EPC ⊕ Ack ⊕ R_r ⊕ R_t) - ← {M_1, R_t, I}
Step 4: Search the database using I: - Found: I = I_{new} {EPC_i, Auk_{new}, Ack_{new}, $data_i$} I = I_{old} {EPC_i, Auk_{old}, Ack_{old}, $data_i$} M_1' = Auk_{new} ⊕ I_{new} ⊕ PRNG(EPC_k ⊕ Ack_{new} ⊕ R_t ⊕ R_r) to authenticate T. - Not Found: termination. - B = $data_i$ ⊕ Rid_k - M_2 = PRNG(Auk_{new} ⊕ R_t) ⊕ Ack_{new} - C = H($data_i$ ⊕ R_r) - Update database values and keys: Auk_{old} = Auk_{new} Auk_{new} = PRNG(Auk_{new}) Ack_{old} = Ack_{new} Ack_{new} = PRNG(Ack_{new}) I_{old} = I_{new} I_{new} = PRNG(Ack_{new} ⊕ I_{new}) - send {B, C, M_2} →	**Step 3:** - A = H(Rid ⊕ R_r) - ← {M_1, R_t, I, A, R_r}	
	Step 5: - Obtain $data_i$ from B - C' = H($data_i$ ⊕ R_r) - send M_2 →	**Step 6:** - Compute M_2' = PRNG(Auk ⊕ R_t) ⊕ Ack - Update tag values and keys
R_r, R_t: Random No. of reader/tag; Auk: Authentication key of tags shared with server; Rid: Reader ID; EPC: Electronic product code of tag; Ack: Access key of tags shared with server; I: Index value of tag; H(): One-way hash function; $data_i$: Secret information of the tag's object.		

Figure 2.17: Hash-based mutual authentication protocol (HBA+) by Chang

Z.Liu et al. [36] proposed variable linear shift-based authentication protocol (VLP) to support the implementation of RFID for the new EPC Gen2v2 standard, satisfy its security features of untraceability and access control, and reduce a tag's read range. In Figure 2.18, the protocol is based on a lightweight encryption function called Variable Linear Feedback Shift Register (VLFSR), which is implemented at the application-specific integrated circuit (ASIC) level. In every session, mutual authentication involves different random numbers from the tag and reader combined with the new secret value SID stored in the database to provide resistance against active attacks.

Server S	**Reader R**	**Tag T**
	Step 1: - Generate random R_r - Send R_r to tag →	
		Step 2: - Generate random R_{t1}, R_{t2} - Secret value = m_j - B_t = VLFSR ($R_{t1}||R_r$, m_j)
		Step 3: - Send to reader B_t, $(R_{t2}||R_{t1}) \oplus SID_j$ ←
	Step 4: - Send to server R_r, B_t, $(R_{t2}||R_{t1}) \oplus SID_j$ and UID ←	
Step 5: Authenticate and Update 1- Find an SID_j match in database based on UID 2- Extract R_{t1}, R_{t2} 3- $(R_{t2}||R_{t1}) \oplus SID_j$ 4- Find m_j from M_j table based on SID_j 5- B_b = VLFSR ($R_{t1}||R_r$, m_j) 6- If $B_b = B_t$, proceed to update • $m_{j+1} = (R_{t2}||R_{t1}) \oplus m_j$ • SID_{j+1} = VLFSR ($R_{t1}||R_{t2}$, m_j) • Store new values in M_j and keep the old in M_{j-1} tables. 7- If $B_b \neq B_t$, find m_j and SID_j from M_{j-1} table and do step3 8- If no match is found, protocol will stop.		
Step 6: - Send to reader VLFSR ($R_{t1}||R_r$, SID_j) →		
	Step 7: - Send to reader VLFSR ($R_{t1}||R_r$, SID_j) →	**Step 8: Authenticate R and S** - Authentication via received msg. - $m_{j+1} = R_{t2}||R_{t1}) \oplus m_j$ - SID_{j+1} = VLFSR ($R_{t1}||R_{t2}$, m_j)
SID: Session secure ID of tag; UID: Unique ID of tag; R_r: Reader random No.; R_{t1}, R_{t2}: Random No. generated by tag; m_j: Secret value used in a session; VLFSR(): Variable LFSR function.		

Figure 2.18: Variable linear shift-based authentication protocol (VLP) by Z.Liu

Another protocol (OMP) is proposed by Niu et al. [37] mainly for passive tag ownership transfer using a lightweight authentication mechanism to support EPC Gen2 standard. Since the ownership transfer is based on transferring the keys, the OMP protocol aims to prove the possession of the shared secret key to a tag and reader without disclosing it using ultra-lightweight permutation operation (Per), as in Figure 2.19. Yet, the protocol has no mechanism to check the freshness of the message that is sent by a legitimate reader.

Server S	**Reader R: K, K_M, EPC, R_{ID1}**	**Tag T: K, K_M, EPC, R_{ID}, IDS**
Mutual Authentication:	1- Generate random rnd_1, rnd_2 of 96 bits 2- $A_i = rnd_{1i} \oplus PRNG(K_i \oplus R_{ID1i}) \oplus PRNG(K_i \oplus R_{ID2i})$ 3- $B_i = rnd_{2i} \oplus PRNG(rnd_{1i} \oplus K_i)$ 4- C_i $PRNG(rnd_{1i} \oplus R_{ID1i}) \oplus PRNG(rnd_{2i} \oplus R_{ID2i})$ 5- Send A_i, B_i, C_i to tag →	6- Extract rnd_1, rnd_2 - $rnd_{1i} = A_i \oplus PRNG(K_i \oplus R_{ID1i}) \oplus PRNG(K_i \oplus R_{ID2i})$ - $rnd_{2i} = B_i \oplus PRNG(rnd_{1i} \oplus K_i)$ - $C'_i = PRNG(rnd_{1i} \oplus R_{ID1i}) \oplus PRNG(rnd_{2i} \oplus R_{ID2i})$ 7- If C = C', reader authenticated - $K_{i+1} = Per(rnd_{1i}, K_i) \oplus K(i+1 \bmod 6)$ - $IDS_{i+1} = Per(rnd_{2i}, K_i) \oplus K_i$ - $D_i = PRNG(K_{i+1} \oplus IDS_{i+1})$, where i = 1 to 6 8- Send D to reader←
	9- Verify D: - $D'_i = PRNG(K_{i+1} \oplus IDS_{i+1})$, where i = 1 to 6 - If D is verified, tag is authenticated	
K: Secret shared key for owners; K_M: Master key to modify K. EPC: Static ID of a tag. R_{ID}: ID of reader owning tag. IDS: Pointer to tag database.		

Figure 2.19: Passive tag ownership authentication protocol (OMP) by Niu

Dass and Om [38] also proposed an efficient authentication protocol (SEAS) that uses lightweight operations and a pseudo-random number generator (PRNG) for a low computational cost. Their scheme is based on a secure channel between the back-end

server and reader, prestored tags' secret (SIDs) in the side of the tag, a one-way hash function of the tag ID in the server side, and rewritable memory with a flag indicator in the server side to update the secret values. Any change to the messages transmitted leads to terminate the communication during the verification to resist security attacks, as shown in Figure 2.20.

Server S	**Reader R**	**Tag T**
	Step 1: - Generate random No. N_R →	
		Step 2: - Generate random No. N_T - V = PRNG ($S \oplus N_R \oplus N_T$) - H = h (ID)
	Step 3: - ← {V, H, N_R, N_T}	- ← {V, H, N_T}
Step 4: Search the database using h(ID): - Not Found: termination - Found: verify V V' = PRNG($S_{new} \oplus N_R \oplus N_T$) Send S_{new} to reader → Flag = 0		
V" = PRNG($S_{old} \oplus N_R \oplus N_T$) Send S_{old} to reader → Flag = 1	**Step 5:** - Reader takes S_{new} or S_{old} - M = PRNG($S_{new, old}$, N_R) - N = PRNG(M) - Send N to tag →	
Step 6: - Flag = 0 → S = S_{new} U = h (S_{new}‖M) S_{old} = S_{new} S_{new} = $S_{new} \oplus$ U - Flag = 1 → S = S_{old} U = h (S_{old}‖M) S_{old} = S_{old} S_{new} = $S_{old} \oplus$ U	- ← send M to server	**Step 6:** - To authenticate reader: Calculate M' = PRNG(S, N_R) Calculate N' = PRNG(M') Verify N' = N - If equal calculate U = h (S‖M') - Update S = S ⊕ U
h(): One-way hash function; N_R, N_T: Random No. generated by reader/tag; S: Secret value of tag; ID: ID pseudonym of tag; S_{new}, S_{old}: Current and old session secrets of tag.		

Figure 2.20: Efficient authentication protocol (SEAS) by Dass and Om

An alternative solution to replace the central database in the RFID system is to use a *serverless* model in which the database server does not maintain a connection with the readers and tags during the communication. Regarding this challenge, Mtita et al. [39] proposed (SAP), a serverless security protocol used for the mass authentication of RFID tags in the presence of untrusted readers. In SAP protocol, the reader and tag do not communicate with the back-end server; instead, they authenticate each other using only ephemeral of the tag's secrets that expire within a given time, as shown in Figure 2.21. Verification and authentication between the reader and tag are done during the authentication phase to exchange the data and generate the session key locally in both tag and reader for their next communication. The protocol has also been proved using the *CryptoVerif* tool [40], which was shown to have low computation overhead and resources.

Server: S	**Reader:** R_j	**Tag:** T_i
Initialization Phase: 2- Generate K_{ij}, $temp_{ij}$, AR_{ij} (access right) for each tag derived from time window and start date $K_{ij} = HMAC_{idi}(W_{sj}\|\|AR_{ij})$ 3- Build lists of authenticated tags L_j for R_j $L_j - \{(temp_{1j}, K_{1j}), (temp_{2j}, K_{2j}),.., (temp_{ij}, K_{ij})\}$ 4- Send L_j, AR_{ij}, W_{sj} to $R_j \rightarrow$	1- Request permission from server S.	- Ti has Timestamp T_{SYS} and id_i
Mutual Authentication Phase:	1- Generate r_j 2- Send to tag A= W_{Sj}, AR_{ij}, $r_j \rightarrow$	3- Generate r_i 4- $H_{ij} = HMAC_{Kij}(r_i\|\|r_j)$ 5- Send to reader B = H_{ij}, $r_i \leftarrow$
	6- Verify $H'_{ij} = HMAC_{Kij}(r_i\|\|r_j)$ If equal: T_i is authenticated If not equal: K_{ij} is not in the list and tag is not authorized 7- Generate timestamp t_j and calculate $V_{ij} = HMAC_{Kij}(r_i\|\|t_j)$ 9- Send to tag C = t_i, $V_{ij} \rightarrow$	10- Verify $V'_{ij} = HMAC_{Kij}(r_i\|\|t_j)$ If equal: R_j is authenticated 11- Update $T_{SYS} = t_j$
	12- Generate session key $K_S = HMAC_{Kij}(t_j\|\|r_i\ W_{Sj})$	13- Generate session key $K_S = HMAC_{K'ij}(t_j\|\|r_i\ W_{Sj})$
T_{SYS}: Tag static timestamp; t_j: Reader timestamp; id_i: Tag ID; K_{ij}: Tag's key; $temp_{ij}$: Temporary tag ID; AR_{ij}: Access rights; W_{sj}: Time window; Ks: Session key; L_j: List of authorized tags; r_i, r_j: Reader/Tag random No.		

Figure 2.21: Serverless security authentication protocol (SAP) by Mtita

2.1.4 Ultra-Lightweight Protocols

As mentioned earlier in this paper, passive tags are small chips with scarce resources that can only support low-cost operations. The goal of ultra-lightweight protocols is to reduce the cost of RFID systems at a minimum and provide strong security for promising future use. In this regard, Sundaresan et al. [5] introduced an ultra-lightweight serverless protocol (STS) using only simple XOR and 128-bit PRNG operations that require less than 2000 gates, three random number generation on the tag, and two message exchanges. In Figure 2.22, the STS protocol mechanism is to use a blind factor to hide the pseudo-random numbers that are used in communication between readers and tags to overcome impersonation attacks. An RFID tag is also able to preserve its location privacy by responding as a noise tag. Moreover, the protocol does not employ a one-way hash function nor any encryption conforming to EPC C1 G2 Standards.

Aggarwal and Das [41] proposed the CHW+ protocol, which is based on a previous version introduced by Y. Chen, Wang, and Hwang (CWH) [42]. In Figure 2.23, the protocol CHW+ solves the problem of full disclosure attack due to the simple XOR operation that is used in the authentication message, which uses the bit rotation and shifting operation on the message before transmission to increase the protocol complexity. CWH+ protocol is resistant to replay attack, forge attack, and DoS with a very efficient computation.

Huang and Li [43] proposed and implemented two improved protocols of RFID mutual authentication based on generating PadGen function in the ISO 18000-6C [44]

Server: t_s	**Reader R**	**Tag T**
Setup Phase: - Stores access list (AL) of all n tags: $h(TID_1, t_{s1}) = id_1, rts_1, ctr_1, ctrmax_1$ $h(TID_n, t_{sn}) = id_n, rts_n, ctr_n, ctrmax_n$ - Establish shared rts between a reader and each tag to be searched. **Search Phase:** - Server is offline.	- Precompute and store id = $h(TID, t_s)$ **Step 1:** 1- Check that ctr <= ctrmax 2- Generate PRN r_r 3- B = $rts_j \oplus id_i$ 4- M1 = $id_i \oplus PRNG (rts_j \oplus r_r)$ 5- M2 = $r_r \oplus B$ 6- Broadcast M1, M2 to all tags → **Step 3:** 1- Extract t_r from M4 2- Verify rts_j = M3 ⊕ PRNG ($id_j \oplus t_r$) 3- If verified, tag is authenticated 4- Update rts_j = M3 ⊕ PRNG ($id_j \oplus t_r$) ctr = ctr + 1	- Stores id = $h(TID, t_s)$. - Stores for each reader: $rts_1, rts_1^{-1}, ctr_1, ctrmax_1, r_{r1}^{-1}$ $rts_m, rts_m^{-1}, ctr_m, ctrmax_m, r_{rm}^{-1}$ - ctr = 0. **Step 2:** 1- B = rts ⊕ id 2- Extract r_r from M2 = B ⊕ r_r 3- Check r_r: - If $r_r = r_r^{-1}$, a replayed msg, exit. - If $r_r \neq r_r^{-1}$, proceed 4- Verify id = M1 ⊕ PRNG (rts ⊕ r_r): - If equal, reader is authenticated - If not equal, repeat using rts^{-1}. - If not equal, respond with λ and exit. 5- If id is verified, check if ctr < ctrmax: - Generate PRN t_r - M3 = rts ⊕ PRNG (id ⊕ t_r) - M4 = $t_r \oplus B$ 6- Update $r_r^{-1} = r_r$ 7- If id is verified using rts: - Update rts^{-1} = rts - rts = PRNG (rts) - ctr = ctr + 1 8- Send M3, M4 to reader ←
AL: Access list for the reader; t_s: Secret key of tag; rts, rts^{-1}: Shared secrets between reader/tag; B: Blind factor to hide PRN; ctr: Counter value; ctrmax: Number of times a reader is pre-authorized to search; TID: Tag ID; id: hashed value of TID; r_r: Random No. of reader in current session.		

Figure 2.22: Ultra-lightweight serverless authentication protocol (STS) by Sundaresan

protocol to protect the memory with a 32-bit access password. The concept of their protocols is to cover up the tag's access password (Apwd) before transmitting the data using a set of 16-bit random numbers such as RTx and RMx. One of the improved schemes, PadGen with XOR (PGX), implements XOR operation between the random

number sets and the PadGen function; the other protocol, PadGen with Mod (PGM), implements a Modulo operation (MOD9) in the eight-bit half of the 16-bit random number set (RTx, RMx) to be used in the PadGen function. Both improved schemes conform to the EPC C1 G2 standard, do not require any hash function or key exchange, do not involve synchronization for hash or key values, and also show better efficiency during implementation. The security level of the MOD scheme is higher due to the low-cost implementation but requires a higher computation cost in PadGen than XOR.

Server S	**Reader R**	**Tag T**
	Step 1: - Generate random r. - s = RID ⊕ r - n1 = weight(RID), n1 = no. of bit value 1 of RID - s' = Rot(s, n1) - Send s' to tag →	
		Step 2: - s = Rot'(s', n1) - Retrieve r = RID ⊕ s - n1 = weight(r) - r' = Rot(r, n1) - r'_{prev} = Rot(r_{prev}, n1) - t1 = (TID ⊕ r'_{prev}) + (r' ⋀ r'_{prev}) - Send t1 to reader ←
	Step 3: - Forward t1, r to server ←	
- n = weight(r) - r' = Rot(r,n) - r'_{prev} = Rot(r_{prev}, n) - Retrieve and verify TID to authenticate tag. - r'' = Rot(r', n) - r''_{prev} = Rot(r'_{prev}, n) - t2 = (TID + r''_{prev}) ⋀ r'' - Update r_{old} = r_{prev}, r_{prev} = r' - Send t2 to tag →		
		Step 4: - Compute r'', r''_{prev} as the server - t'2 = (TID + r''_{prev}) ⋀ r'' - Verify t'2 = t2 - Update r_{prev} = r'
RID: Reader ID; Rot(): Rotation function; TID: Tag ID; Weight(r): number of 1's in the binary string shifts r to the left for n bits.		

Figure 2.23: Improved authentication protocol (CWH+) by Aggarwal and Dass

Huang and Jiang [45] proposed an ultra-lightweight reader–tag mutual authentication protocol (MACC) based on Chien and Chen's protocol [46] to overcome forge attacks, DoS, and forward security attacks. Although the improved scheme uses only lightweight operations such as RNG, PRNG, and XOR, it involves an exhaustive search in the database for tag pseudo-IDs in every session that leads to computational overhead, as shown in Figure 2.24. It also fails to resist tracking attacks.

Server S	Reader R	Tag T
	Step 1: - Generate r_1 - $V_R = h(RID_j \oplus r_1)$ - Send to tag $r_1 \rightarrow$	**Step 2:** - Generate r_2 - $M_1 = N_i \oplus r_2$ - $M_2 = P(EPC_i \| r_1 \| r_2 \| K_i)$ - Send to reader $(M_1, M_2, PID_i) \leftarrow$
	Step 3: - Send to server $(M_1, M_2, PID_i, r_1, V_R)$	
Step 4: - Verify V_R using reader ID and r_1 - Search database for tag PID_i - Verify M_1 as: $r_2 = M_1 \oplus N_i^{old}$ OR $r_2 = M_1 \oplus N_i^{new}$ - Verify M_2 as: $M_2 = P(EPC_i \| r_1 \| r_2 \| K_i^{old}$ or $K_i^{new})$ - $M_3 = P(EPC_i \| r_2 \| N_i^x \| K_i^x)$ x = old or new - Send M_3 to reader $\rightarrow$	**Step 5:** - Forward M_3 to tag $\rightarrow$	**Step 6:** - Verify $M_3 = P(EPC_i \| r_2 \| N_i \| K_i)$ - $N_i = P(N_i \oplus r_2)$ - $K_i = P(K_i \oplus r_2)$ - $PID_i = P(PID_i \oplus r_2)$
- If x = new, proceed to update $N_i^{old} = N_i^{new}$, $N_i^{new} = P(N_i^{new} \oplus r_2)$ $K_i^{old} = K_i^{new}$, $K_i^{new} = P(K_i^{new} \oplus r_2)$ $PID_i^{old} = PID_i^{new}$, $PID_i^{new} = P(PID_i^{new} \oplus r_2)$		
P: Access key with reader; PID: Pseudonym ID of tag; N_i: Nonce; EPC: Electronic product code for the tag; K_i: Authentication key.		

Figure 2.24: Ultra-lightweight mutual authentication protocol (MACC) by Huang and Jiang

Huang and Jiang [45] proposed another mutual authentication protocol (MACD) based on Chen and Deng's scheme [47] to overcome forge attacks, DoS, replay attacks, and mainly the tag identification time. It is shown in Figure 2.25 that the MACD protocol uses ultra-lightweight operations and achieves a lower communication cost between tag and reader than the other improved scheme, MACC.

Server S	Reader R	Tag T
	Step 1: - Generate r_1 - Send r_1 to tag →	**Step 2:** - Generate r_2 - $A = r_1 \oplus r_2 \oplus P_i$ - $B = P(P_i \| r_1) \oplus r_2$ - $M_1 = CRC(EPC_i \| A \| B \| K_i)$ - Send to reader (M_1, B) ←
Step 4: - Search database for EPC_i match - $r_2 = B \oplus (P_i^{old} \| r_1)$ OR $B \oplus (P_i^{new} \| r_1)$ - $A' = r_1 \oplus r_2 \oplus P_i^{old}$ or P_i^{new} - Verify $M_1 = CRC(EPC_i \| A' \| B \| K_i^{old}$ or $K_i^{new})$ - $M_2 = CRC\ (EPC_i \| r_2 \| P_i^x \| K_i^x)$ x = old, new - Send M_2 to reader →	**Step 3:** - Send to server (M_1, B, r_1)	
- If x = new, proceed to update $P_i^{old} = P_i^{new}$, $P_i^{new} = P(P_i^{new} \oplus r_2)$ $K_i^{old} = K_i^{new}$, $K_i^{new} = P(K_i^{new} \oplus r_2)$	**Step 5:** - Send M_2 to tag →	**Step 6:** - Verify $M_2 = CRC\ (EPC_i \| r_2 \| P_i \| K_i)$ - $P_i = P(P_i \oplus r_2)$ - $K_i = P(K_i \oplus r_2)$
P: Access key with reader; K_i: Authentication key; N_i: Nonce; EPC: Electronic product code for the tag; CRC(): Cyclic redundancy check function.		

Figure 2.25: Mutual authentication protocol (MACD) by Huang and Jiang

Considering the complexity of the authentication protocol, Hopper and Blum proposed the first HB protocol to identify unaided humans to computers [48]. Many authors adopted the idea of HB protocol to identify tags in RFID networks. HB family protocols are based on the hard problems of Learning Parity with Noise (LPN), which involves the calculation of inner products of binary vectors and Bernoulli noise bit generation [49]. In this regard, Lin and Song [50] proposed HBROT, which is one of the latest HB protocols that produce the key in each authentication round using the rotation function. The protocol is considered to be secure against most of the RFID attacks.

Another improvement of the HB protocol is proposed by Juels and Weis [51] as (HB+) to overcome the weaknesses of the original HB. The HB+ protocol involves two secret keys, x, and y, which are used with shared blind vectors between the reader and tag. The reader and tag verify the values that are computed to perform mutual authentication. Later, the protocol is reported by Gilbert et al. [52] to be vulnerable to the man-in-the-middle attack (MIM). Hence, Ouaskou et al. [53] proposed a variant of HB protocol based on Permutation function (HBPER). The protocol performs a permutation of the keys x,y during each round of the protocol to update the value of the keys, as shown in Figure 2.26. This method secures the protocol against the MIM attack that is reported in the HB+ protocol, although both protocols HB+ and HBPER almost have the same complexity.

Reader R	**Tag T**
$x = x_{k-1}, \ldots, x_1, x_0$ $y = y_{k-1}, \ldots, y_1, y_0$ **Step 1:** - Generate random challenge (a) - Send (a) to T →	$x = x_{k-1}, \ldots, x_1, x_0$ $y = y_{k-1}, \ldots, y_1, y_0$ **Step 2:** - Compute y = Per(y,a) - Compute x = Per(x,a) - Compute z = a . x ⊕ v (v = noise bit; v = 1 with probability of η) - Send z to R ←
Step 3: - Compute y = Per(y,a) - Compute x = Per(x,a) - Verify z = a . x	
x: Shared key by tag and reader of k-bit; k: Length of secret keys; y: Shared key by tag and reader of k-bit; η = noise level ∈]0,1/2[.	

Figure 2.26: A variant of hb protocol based on permutation function (HBPER) by Ouaskou

2.2 Analysis and Security Evaluation

In this section, we compare the different protocols in terms of computation, security requirements, and attacks resistance. Table 2.2.1 demonstrates the different operations computed by the tag in each protocol and the communication overhead based on the number of transmitted messages between tag and reader.

2.2.1 Comparison of Computation Cost

We denote T_{ENC}, T_{DEC}, T_{PRNG}, T_{RNG}, T_{SMUL}, T_{XOR}, T_{CH}, T_{H}, T_{CRC}, T_{ROT}, T_{SHIFT}, T_{ITER}, T_{BIT}, T_{SPR}, T_{PER}, T_{MOD}, T_{VLFSR} as the computation cost for encryption, decryption, pseudo-random number generator, random number generator, scalar multiplication, XOR, cryptographic hash, one-way hash function, cyclic redundancy check, rotation, shifting, iteration, bitwise operation, spreading, permutation, modulo, variable linear shift register function, respectively. Tag overhead is classified based on the cryptographic level of operations used in the protocol: *high* for symmetric key cryptography and scalar multiplication, *medium* for a one-way hash function, and *low* for other bitwise operations and random number generators. The passes are designated for the number of messages sent by a reader or a tag.

2.2.2 Comparison of Security Threats

Protocols resistance to different RFID threats is presented in Table 2.2, where we denote ST1 for a replay attack, ST2 for a man-in-the-middle attack (MITM), ST3 for eavesdropping, ST4 for an impersonating attack, ST5 for traceability, ST6 for desynchronization , ST7 for denial of service (DoS), and ST8 for other types of attack.

Table 2.2.1 Comparison of the Computation Cost on Tag

Protocol	Operations	Tag Passes	Reader Passes	Tag Overhead
SB-A [9]	$1\ T_{ENC} + 2\ T_{DEC} + 2\ T_{PRNG}$	2	3	High
SB-B [9]	$2\ T_{ENC} + 2\ T_{DEC} + 2\ T_{PRNG}$	2	3	High
EMA [6]	$2\ T_{SMUL} + 2\ T_{CH}$	2	1	High
ECU [11]	$2\ T_{SMUL} + 2\ T_{CH}$	1	1	High
SPA [12]	$4\ T_{SMUL} + 1\ T_{CH}$	1	1	High
PII [13]	$4\ T_{SMUL} + 3\ T_{CH}$	1	1	High
RUND [14]	$2\ T_H$ OR $1\ T_{ENC} + 1\ T_{PRNG}$	1	2	High
IECC [15]	$2\ T_{SMUL} + 2\ T_H$	1	2	High
EECC [16]	$2\ T_{SMUL} + 2\ T_H$	1	2	High
RBAC [18]	$2\ T_{ENC} + 2\ T_{DEC} + 1\ T_{PRNG}$	2	2	High
DRAP [30]	$1\ T_{ENC} + 3\ T_{XOR} + 3\ T_H + 1\ T_{RNG} + 2\ T_{PRNG}$	1	2	High
NRS [19]	$10\ T_{XOR} + 3\ T_H$	4	5	Medium
NRS+ [8]	$10\ T_{XOR} + 6\ T_H$	4	5	Medium
NRS++ [20]	$8\ T_{XOR} + 4\ T_H$	1	2	Medium
ACSP [21]	$3\ T_{XOR} + 7\ T_H + 4\ T_{CRC}$	1	4	Medium
ACSP+ [22]	$4\ T_{XOR} + 8\ T_H$	2	4	Medium
ACSP++ [20]	$6\ T_{XOR} + 8\ T_H$	1	2	Medium
MASS [28]	$4\ T_{XOR} + 2\ T_H + 1\ T_{RNG}$	1	2	Medium
EP-UAP [29]	$2\ T_H + 1\ T_{RNG}$	1	2	Medium
GUPA [31]	$2\ T_H + 3\ T_{PRNG} + 19\ T_{BIT}$	3	3	Medium
HBA [34]	$6\ T_{XOR} + 2\ T_H + 1\ T_{RNG} + 4\ T_{MOD}$	1	2	Medium
VLP [36]	$2\ T_{XOR} + 2\ T_{RNG} + 3\ T_{BIT} + 2\ T_{VLFSR}$	1	2	Medium
SEAS [38]	$1\ T_{XOR} + 2\ T_H + 1\ T_{RNG} + 3\ T_{PRNG} + 1\ T_{BIT}$	1	2	Medium
SAP [39]	$2\ T_H + 2\ T_{RNG}$	1	2	Medium
LAP [23]	$2\ T_{XOR} + 1\ T_{RNG} + 2\ T_{PRNG} + 1\ T_{ROT} + 1\ T_{SHIFT}$	2	2	Low
Flyweight [26]	$5\ T_{PRNG}$	3	3	Low
FTest [32]	$1\ T_{XOR} + 3\ T_{CRC}$	3	2	Low
ACS [33]	$3\ T_{XOR} + 2\ T_{ITER} + 1\ T_{SPR}$	1	2	Low
HBA+ [35]	$7\ T_{XOR} + 1\ T_{RNG} + 5\ T_{PRNG}$	1	2	Low
OMP [37]	$12\ T_{XOR} + 6\ T_{PRNG} + 2\ T_{PER}$	1	1	Low
STS [5]	$7\ T_{XOR} + 3\ T_{PRNG}$	1	1	Low
CWH+ [41]	$2\ T_{XOR} + 5\ T_{ROT} + 1\ T_{SHIFT} + T_{BIT}$	1	1	Low
PGX [43]	$8\ T_{XOR} + 2\ T_{RNG}$	2	2	Low
PGM [43]	$4\ T_{XOR} + 2\ T_{RNG} + 32\ T_{MOD}$	2	2	Low
MACC [45]	$6\ T_{XOR} + 5\ T_{PRNG}$	1	2	Low
MACD [45]	$5\ T_{XOR} + 3\ T_{PRNG} + 1\ T_{CRC}$	1	2	Low
HBROT [50]	$1\ T_{RNG} + 2\ T_{ROT} + 1\ T_{XOR} + 1\ T_{BIT}$	1	1	Low
HBPER [53]	$1\ T_{RNG} + 2\ T_{PER} + 1\ T_{XOR} + 1\ T_{BIT}$	1	1	Low

T_{ENC}: encryption, T_{DEC}: decryption, T_{PRNG}: pseudo-random number generator, T_{RNG}: random number generator, T_{SMUL}: scalar multiplication, T_{XOR}: XOR, T_{CH}: cryptographic hash, $T_{H:}$ one-way hash function, T_{CRC}: cyclic redundancy check, T_{ROT}: rotation, T_{SHIFT}: shifting, T_{ITER}: iteration, T_{BIT}: bitwise operation, T_{SPR}: spreading, T_{PER}: permutation, T_{MOD}: modulo, T_{VLFSR}: variable linear shift register function.

We found that most of the recently proposed protocols do not pay close enough attention to DoS, MITM, and eavesdropping attacks, while most of the protocols consider the system security against replay, impersonate, traceability, and desynchronization attacks. Certainly, protocols [35],[15],[16],[20],[31],[50, 53] are strongly resistant to all of the major attacks.

2.2.3 Comparison of Security Requirements

Security requirements for an RFID system should be satisfied with the system to defend against the attacks mentioned in this paper. Table 2.2.3 compares the security requirements in each protocol, which includes mutual authentication (SR1), confidentiality (SR2), message integrity (SR3), privacy (SR4), forward secrecy (SR5), backward secrecy (SR6), tag anonymity (SR7), and conforming to EPC standards (SR8). We found that most of the protocols fully considered mutual authentication, privacy, and data protection, while backward secrecy is given the least attention, and should be more considered in future work. However, Niu et al. [37] and X. Chen [20] completely satisfied all of the security requirements in their protocol.

Since the RFID passive tag has limited resources to compute complex operations, the heavyweight and simple-weight protocols are not feasible for practical implementation. However, lightweight and ultra-lightweight protocols use only simple operations within the tag computation limits and show the lowest tag overhead level. Lightweight and ultra-lightweight protocols are considered the most suitable for the current applications. Another vital aspect when considering the appropriate RFID protocol is the security resistance to the attacks. We found out that Chang et al. [35],

Table 2.2.2: Comparison of Various System Requirements

	ST1	ST2	ST3	ST4	ST5	ST6	ST7	ST8
SB-A [9]	Y	Y	Y	Y	Y	Y	*	Cloning
SB-B [9]	Y	Y	Y	Y	Y	Y	*	Cloning
ECU [11]	Y	Y	*	Y	Y	*	*	*
SPA [12]	N	*	*	N	Y	*	*	*
EMA [6]	Y	*	*	N	N	*	*	*
PII [13]	Y	*	*	Y	Y	*	*	*
RUND [14]	Y	*	*	Y	Y	Y	Y	*
IECC [15]	Y	Y	Y	Y	Y	Y	Y	Cloning
EECC [16]	Y	Y	Y	Y	Y	Y	Y	Spoofing
RBAC [18]	Y	*	*	Y	Y	*	Y	*
NRS [19]	N	Y	N	N	N	N	N	*
NRS+ [8]	N	Y	Y	N	N	N	N	*
NRS++ [20]	Y	Y	Y	Y	Y	Y	Y	*
ACSP [21]	N	N	N	N	N	N	N	Counting
ACSP+ [22]	N	*	*	N	Y	Y	N	Counting
ACSP++ [20]	Y	Y	Y	Y	Y	Y	Y	Counting
LAP [23]	Y	*	*	N	N	N	Y	*
Flyweight[26]	Y	Y	Y	Y	Y	Y	*	*
MASS [28]	N	N	N	N	Y	N	*	*
EP-UAP [29]	N	Y	Y	N	Y	*	*	*
DRAP [30]	Y	*	*	Y	Y	Y	Y	Y
GUPA [31]	Y	Y	Y	Y	Y	Y	Y	DoP
FTest [32]	Y	Y	Y	Y	Y	*	*	Counterfeit +Collision
ACS [33]	Y	Y	Y	Y	Y	*	*	Counterfeit +Collision
HBA [34]	N	Y	Y	Y	Y	Y	N	Brute + Counterfeit
HBA+ [35]	Y	Y	Y	Y	Y	Y	Y	Brute for
VLP [36]	Y	Y	Y	*	Y	Y	*	*
OMP [37]	N	*	*	Y	Y	Y	Y	*
SEAS [38]	Y	Y	*	Y	Y	Y	Y	*
SAP [39]	Y	*	Y	Y	Y	*	*	*
STS [5]	Y	*	*	Y	Y	Y	Y	*
CWH+ [41]	Y	*	Y	Y	*	Y	*	Disclosure
PGX [43]	Y	Y	Y	Y	N	Y	*	Cloning
PGM [43]	Y	Y	Y	Y	N	Y	*	Cloning
MACC [45]	Y	Y	*	Y	N	Y	Y	*
MACD [45]	Y	Y	*	Y	Y	Y	Y	*
HBROT [50]	Y	Y	Y	Y	Y	Y	Y	*
HBPER [53]	Y	Y	Y	Y	Y	Y	Y	*

ST1: replay attack, ST2: man-in-the-middle, ST3: eavesdropping, ST4: impersonate attack, ST5: traceability, ST6: desynchronization, ST7: DoS, ST8: other types of attack, Y: satisfied, N: not satisfied. *: not applicable

Table 2.2.3: Comparison of the Security Requirements

	SR1	SR2	SR3	SR4	SR5	SR6	SR7	SR8
SB-A [9]	Y	Y	Y	Y	Y	*	Y	N
SB-B [9]	Y	Y	Y	Y	Y	*	Y	N
ECU [11]	N	Y	Y	Y	Y	*	Y	N
SPA [12]	*	*	*	*	N	*	*	*
EMA [6]	*	*	*	*	N	*	*	*
PII [13]	*	*	*	*	N	*	*	*
RUND [14]	Y	Y	Y	Y	Y	*	Y	N
IECC [15]	Y	Y	Y	Y	Y	Y	Y	N
EECC [16]	Y	Y	Y	Y	Y	Y	Y	N
RBAC [18]	Y	Y	Y	Y	*	*	Y	N
NRS [19]	N	Y	N	N	N	N	N	Y
NRS+ [8]	N	Y	Y	N	N	N	N	Y
NRS++ [20]	Y	Y	Y	Y	Y	Y	Y	Y
ACSP [21]	Y	N	N	N	N	N	N	Y
ACSP+ [22]	Y	Y	Y	*	N	Y	*	Y
ACSP++ [20]	Y	Y	Y	Y	Y	Y	*	Y
LAP [23]	Y	Y	Y	N	Y	*	N	Y
Flyweight [26]	Y	Y	Y	Y	Y	Y	Y	Y
MASS [28]	Y	Y	N	*	Y	*	*	Y
EP-UAP [29]	N	Y	Y	Y	*	*	Y	Y
DRAP [30]	Y	*	*	Y	*	*	Y	Y
GUPA [31]	Y	Y	Y	Y	Y	*	Y	Y
FTest [32]	N	Y	Y	Y	Y	*	Y	Y
ACS [33]	Y	*	*	Y	*	*	Y	Y
HBA [34]	Y	Y	Y	Y	Y	*	Y	Y
HBA+ [35]	Y	Y	Y	Y	Y	*	Y	Y
VLP [36]	Y	Y	Y	Y	Y	*	Y	Y
OMP [37]	Y	Y	Y	Y	Y	Y	Y	Y
SEAS [38]	Y	Y	Y	Y	Y	*	Y	Y
SAP [39]	Y	Y	Y	*	*	*	*	Y
STS [5]	Y	Y	Y	Y	Y	*	Y	Y
CWH+ [41]	Y	Y	Y	*	Y	*	*	Y
PGX [43]	Y	*	*	N	*	*	N	Y
PGM [43]	Y	*	*	N	*	*	N	Y
MACC [45]	Y	Y	Y	N	Y	*	N	Y
MACD [45]	Y	Y	Y	Y	Y	*	Y	Y
HBROT [50]	Y	Y	Y	Y	Y	*	*	Y
HBPER [53]	Y	Y	Y	Y	Y	*	*	Y

SR1: mutual authentication, SR2: confidentiality, SR3: message integrity, SR4: privacy, SR5: forward secrecy, SR6: backward secrecy, SR7: tag anonymity, SR8: conforming to EPC standard, Y: satisfied, N: not satisfied, *: not applicable.

Farash [15], Zhang and Qi [16], X. Chen et al. [20], Liu et al. [31], Lin and Song [50], and Ouaskou et al. [53] protocols successfully resist all of the major attacks. Although the other protocols could not resist all of the attacks, they could perform better than the fully secure protocols in term of computation cost; examples include the protocols presented in Farash [15], Zhang and Qi [16], X. Chen et al. [20], and Liu et al. [31],

which have high computation overhead on the tag side. We encourage researchers to pay attention to the forward and backward security since most protocols do not reflect on these two types of attacks. Finally, maintaining the basic security requirements for an RFID system is required to achieve protection against the mentioned attacks in this literature. We assess that only the protocols of Niu et al. [37] and X. Chen et al. [20] satisfy all of the security requirements to maintain the system in a stable and available state. Even though this review shows security variation among the reviewed protocols, each one could still be a preference over others, depending on the requirements of the application in hand.

CHAPTER 3: PROPOSED SERVERLESS RFID AUTHENTICATION MODEL

3.1 Network Model

RFID passive tags are distributed in an area of interest and attached to mobile objects, i.e. cars. All the tags have the same resources and computational capabilities. The passive tag has no power source and gets activated based on the electromagnetic waves that are sent from the reader at the beginning of the communication. The RFID reader is a scanning device that is either in a fixed position or mobile handheld. It has more resources and computational capability than the passive tag. It collects the tag information such as the Electronic Product Code (EPC) [3] that is a 96-bit string of data contains the tag identity, organization, protocol, product type, and owner. The reader reports the scanned information to the database server. The Server is a centralized database device with a computer program that delivers, stores and manages all the information of the reader and tag. The reader interrogates the tag in the range by sending a challenging request signal to start the communication. The tag, on the other hand, responds to the reader's request based on the approved protocol to verify its legitimate identity. The reader forwards the tag's response to the server to search for the correct information of the tag in the database. The server supports the reader to authenticate the

tag to start a secure channel between the reader and the tag for their further communication. In addition, the tag also uses the approved protocol to verify the reader's identity to avoid compromising the secret information or location of the tag.

3.2 Serverless Model

The server role is eliminated in the proposed serverless model of RFID. The backend server is not available during the communication between the reader and the tag. The reader and tag should be able to verify each other and process the authentication messages successfully while the server is offline. Since the passive tag is considered a low constraint device with scarce resources, the transmitted message between the reader and the tag should carry simple operations within the capability of the tag to perform. Therefore, we consider the elliptic curve cryptography that can be operated by the passive tag to exchange the secret keys. We employed the elliptic curve key agreement based on the discrete log problem in Diffie-Hellman algorithm [54] that allows the reader and the tag to establish a shared key from their public and private keys through an insecure channel to encrypt the transmitted messages. The elliptic curve is a plane curve over a finite field that contains points satisfying the following equation:

$$\boldsymbol{y^2 = x^3 + ax + b} \tag{1}$$

The protocol uses the multiplicative group of integers modulo $\boldsymbol{P}$, and $\boldsymbol{G}$ as a primitive root modulo $\boldsymbol{P}$, where $\boldsymbol{P}$ is prime. The reader and the tag choose random integers a, b respectively as their private keys and compute their public keys as the following:

$$\boldsymbol{A = G^a \bmod P} \quad (2)$$

$$\boldsymbol{B = G^b \bmod P} \quad (3)$$

The values of **A** and **B** are exchanged between the reader and the tag. Then, the reader computes the shared secret *s* using the receiver ***B,*** and ***G***, ***P*** as the following:

$$\boldsymbol{s = B^a \bmod P} \quad (4)$$

The tag also computes the shared secret *s* using the received ***A***, and ***G***, *P* as the following:

$$\boldsymbol{s = A^b \bmod P} \quad (5)$$

As a result, both the reader and the tag end up calculating the same value as their shared secret keys because the modulo rules satisfy the following:

$$\boldsymbol{A^b \bmod P = G^{ab} \bmod P = G^{ba} \bmod P = B^a \bmod P} \quad (6)$$

which also means:

$$\boldsymbol{(G^a \bmod P)^b \bmod P = (G^b \bmod P)^a \bmod P} \quad (7)$$

Based on the points ***P*** and ***G***, the resulted shared secret can take any value between **1** and ***P*-1** that satisfies the following condition:

$$\boldsymbol{1 \leq s \leq P - 1}$$

The security of the elliptic curve algorithm lies in the complexity of computing the original values of public and private keys to obtain the secret key.

3.3 Communication Model

In this section, we present the communication model between the RFID entities.

3.3.1 Setup Phase

This phase handles transferring the necessary data and values from the database server to the reader and the tag. The server, the reader, and the tag share by manufacturer: the elliptic curve point generator and the server public key. The tag by default stores its random identifier that is updated every session to protect the real identity of the tag. The setup phase is also considered a renewal phase such that the reader and the tag request new values to start a new communication session. The renewal phase is necessary when the timestamp expires, or any secret value is compromised to an unauthorized party.

Table 3.3.1: Protocol Notations

P	Point generator of G
G	An additive group of prime order q on an elliptic curve
y_{prv}, r_{prv} , t_{prv}	Private keys of server, reader, tag
Y_{pub}, R_{pub} , T_{pub}	Public keys of server, reader, tag
X_i	Tag identifier
Tag_i	tag ID, group ID, Timestamp
Gk_j , Gk_j^{old}	Current and old values for Group ID
$List_k$	List of tags share the same group ID

Unlike the currently available RFID protocols, the setup phase in *SLEC* protocol is assumed to be insecure and functions as the following steps:

1) The reader and the tag generate random numbers r_{prv1}, t_{prv1} respectively, where $r, t \, \epsilon Z_q$, then compute their public key using the private keys and the point generator as:

$$\boldsymbol{R_{pub1} = r_{prv1} * P} \quad (8)$$

$$\boldsymbol{T_{pub1} = t_{prv1} * P} \quad (9)$$

2) The reader and the tag compute the server secret message of M_1, M_2 respectively using their private keys r_{prv1}, t_{prv1} and the stored public key of the server Y as the following:

$$\boldsymbol{M_1 = r_{prv1} * Y} \tag{10}$$

$$\boldsymbol{M_2 = t_{prv1} * Y} \tag{11}$$

3) The reader and tag send the computed server shared secret M_1, M_2 to the server, then the server obtains the public keys of both reader and tag as:

$$\boldsymbol{R'_{pub1} = y^{-1} * M_1} \tag{12}$$

$$\boldsymbol{T'_{pub1} = y^{-1} * M_2} \tag{13}$$

4) The server, in turn, computes the shared secret for each reader and tag for further communication with the reader and the tag as the following:

$$\boldsymbol{M'_1 = y_{prv} * R == r_{prv1} * Y = M_1} \tag{14}$$

$$\boldsymbol{M'_2 = y_{prv} * T == t_{prv1} * Y = M_2} \tag{15}$$

5) The server generates and stores the following information for each tag:

- Random X_i as tag identifier.
- Timestamp Ts.
- Group ID Gk.

  such that $Tag_i = \{X_i, GK, Ts\}$

6) The server then generates a list of tags for each reader contains a group of tags that share the same group ID.

7) Further, the server sends the reader the tag list as M_3, and sends the tag its information as M_4:

$$\boldsymbol{M_3 = List_k + h(R'_{pub1}, M'_1)} \quad (16)$$

$$\boldsymbol{M_4 = Tag_i + +h(T'_{pub1}, M'_2)} \quad (17)$$

8) The reader computes and verifies the hash value to obtain the list of tags. The tag also validates the hash value to receive the tag information. The server current public key is not shared during the communication, so only the legitimate reader and tag that have the real server public key will be able to compute and verify the hash value to obtain the messages sent by the server

3.3.2 Authentication Phase

When the setup phase is completed successfully, each reader will have a list of tags that have: tag ID, group ID, a timestamp for each tag, and the tag will have: tag ID, group ID, and timestamp. The communication starts with mutual authentication between the reader and the tag as the following steps:

1) The reader generates a random number r_{prv2}, where $r \epsilon Z_q$ then computes its public key using the private keys and the point generator as:

$$\boldsymbol{R_{pub2} = r_{prv2} * P} \quad (18)$$

2) The reader computes the M_1, and M_2 as the following:

$$\boldsymbol{M_1 = h(Gk_j)} \quad (19)$$

$$\boldsymbol{M_2 = R_{pub2} \oplus Ts} \quad (20)$$

Then, the reader sends M_1, and M_2 to the tag.

3) The tag will process four steps:

- Validate $M_1 = h(Gk_j)$ to verify the intended group using the current or the old value of *Gk*. Based on the group verification, the tag generates a random number t_{prv2}, where $t \epsilon Z_q$ and computes its public key as:

$$\boldsymbol{T_{pub2} = t_{prv2} * P} \quad (21)$$

- Obtain the reader public key from M_2

$$\boldsymbol{R_{pub2} = (M_2 \oplus Ts) - Y} \quad (22)$$

- Compute the secret share key with the reader using the reader obtained public key

$$\boldsymbol{M_3 = t_{prv2} * R_{pub2}} \quad (23)$$

- Compute the authentication message M_4 and sends it to the reader

$$\boldsymbol{M_4 = h(X_i, R_{pub2}, T_{pub2}, Gk_j)} \quad (24)$$

- Update the values of the tag *ID*, Gk_j^{old}, and Gk_j

$$\boldsymbol{X_i = PRNG(X_i)} \quad (25)$$

$$\boldsymbol{Gk_j{}^{old} = Gk_j} \quad (26)$$

$$\boldsymbol{Gk_j = PRNG(Gk_j)} \quad (27)$$

4) The reader extracts T'_{pub2} from the received message and verifies the hash value of M_4 to authenticate the tag

$$\boldsymbol{T'_{pub2} = r_{prv2}^{-1} * M_3} \quad (28)$$

5) The reader computes M_5 and sends it to the tag; then, updates the values of the tag *ID*, Gk_j,

$$\boldsymbol{M_5 = h(X_i, R_{pub2}, T'_{pub2})} \quad (29)$$

$$\boldsymbol{X_i = PRNG(X_i)} \quad (25)$$

$$\boldsymbol{Gk_j = PRNG(Gk_j)} \quad (27)$$

6) The tag verifies M_5 to authenticate the reader

3.3.3 Recovery Phase

In an event where any value of the communication is compromised, the tag or the reader can renew the communication values from any server checkpoint during the transportation route. The recovery phase is similar to the security setup phase presented in this chapter. The tag and reader will exchange their newly generated public keys using the server's public key stored in their memory. This will allow the reader and tag to be retrieved back into the network with new values.

CHAPTER 4: METHODOLOGY

4.1 Experiment Design

In our *SLEC* protocol, we created an RFID network with a dynamic size that the number of readers and tags can be increased or decreased. We included one server, five readers, and twenty tags that are placed in objects such as cars. The distance range between the tags and readers is initially assumed to be a few meters based on the reading range of Electronic Product Code Class1 generation 2 of RFID passive tags [3]. The server initializes a database table to store all the readers and tags unique IDs. The readers are placed in fixed positions such as poles along the route of the mobile tags. Before the car departs the dealership inventory, the setup phase is executed, and all the values are stored in the tag and readers. During the tag movement, the reader can scan the tag in the car to perform the mutual authentication and thus, obtain the required information of the tag.

4.2 Serverless Authentication Based on Elliptic curve Algorithm

The proposed protocol is implemented based on the following algorithm:

Algorithm 1 SLEC

Input parameters: minimum value for server_public_key (Y), point generator (P)
tag random identifier (X_i) in server

Server:	Reader:	Tag:
Setup: **Step2:** - extract R_{pub1}, T_{pub1} from M_1, M_2 $R'_{pub1} = y^{-1} * M_1$ $T'_{pub1} = y^{-1} * M_2$ - generate Tag_i: $[X_i, Ts, Gk_j]$ - create $TagList_k$: $[Tag_i, .., Tag_n$ $M_3 = TagList_i + h(R'_{pub1}, M_1)$ $M_4 = Tag_i + h(T'_{pub1}, M_2)$ - send M_3 to reader → - send M_4 to tag →	**Step1:** - select random $r_{prv1} \epsilon Z_q$ - $R_{pub1} = r_{prv1} * P$ - $M_1 = r_{prv1} * Y$ - Forward M_1, M_2 to server ← Step3: - verify the hash value and extract $TagList_i = M_3 - h(R'_{pub1}, M_1)$	- select random $t_{prv1} \epsilon Z_q$ - $T_{pub1} = t_{prv1} * P$ - $M_2 = t_{prv1} * Y$ - Send M_2 to reader ← ¶ - verify the hash value and extract $Tag_i = M_4 - h(T'_{pub1}, M_2)$
Authentication Phase:	**Step1:** - select random $r_{prv2} \epsilon Z_q$ - $R_{pub2} = r_{prv2} * P$ - $M_1 = h(Gk_j)$ - $M_2 = R_{pub2} \oplus Ts$ - send M_1, M_2 to tag **Step3:** - Extract $T'_{pub2} = r_{prv2}^{-1} * M_3$ - Validate M_4 to authenticate tag - $M_5 = h(X_i, R_{pub2}, T'_{pub2})$ - Send M_5 to tag → - Update: $X_i = PRNG(X_i)$ $Gk_j = PRNG(Gk_j)$	**Step2:** - Validate M_1 to verify the group - select random $t_{prv2} \epsilon Z_q$ - $T_{pub2} = t_{prv2} * P$ - extract $R'_{pub2} = M_2 \oplus Ts$ - $M_3 = t_{prv2} * R'_{pub2}$ - $M_4 = h(X_i, R'_{pub2}, T_{pub2}, Gk_j)$ - Send M_3, M_4 to reader ← - Update: $X_i = PRNG(X_i)$ $Gk_j^{old} = Gk_j$ $Gk_j = PRNG(Gk_j)$ **Step4:** - Validate M_5 to authenticate reader

CHAPTER 5: RESULTS AND ANALYSIS

In this section, we present the system performance and security analysis of the protocol *SLEC*. We also compare the *SLEC* protocol to other serverless protocols. The security of SLEC mainly depends on the public key of the main server, which is securely disseminated to all readers and tags. Further, the setup and authentication phases can then be executed through an insecure network to maintain the system requirements and defend the security threats.

5.1 Analysis of System Requirements

The *SLEC* protocol maintains the system requirements that are necessary to create a secure and reliable RFID system such as mutual authentication, confidentiality, integrity, privacy, forward secrecy, anonymity, and availability.

1) Mutual Authentication

The protocol allows both the reader and the tag to perform a mutual authentication since only the legitimate tag can extract the public key of the reader from the message M_2. Besides, only the legitimate reader can calculate the hash value in the message M_5 to prove its identity to the tag. As a result, mutual authentication is satisfied.

2) Privacy and Confidentiality

The transmitted message is confidential because the authentication messages are secured by a hash value that can only be computed by an authorized entity using their secret keys. The privacy of the tag is satisfied as the secret information is protected and not transmitted in the clear.

3) Message Integrity

The message integrity factor is also satisfied because the messages are combined with a digital signature of the sender.

4) Forward and Backward Secrecy

The reader and the tag generate new secret values in every authentication session to avoid tracking or obtaining any secret values from any successful authentication session. Thus, an adversary cannot perform a successful authentication from any previous or expired sessions or anticipate the following authentication messages.

5) Anonymity

The EPC of the tag is not used in the protocol, but only the tag random identifier that is updated every session. As a result, the private information stored in the tag is kept secret.

6) Availability

The protocol provides a recovery mechanism to maintain system availability. In an event where any tag or any secret value of the communication is compromised, the

system can recover the tag by sending new values to the tag during the recovery phase to perform a new authentication session as long as the public key of the server remains secret. Otherwise, a new setup phase is required to feed the tag with a new public key for the server.

7) Scalability

We introduce the concept of tag grouping in *SLEC*. We combine a number of tags into a group that shares the same group ID with all the tags, but each tag in the group has a unique tag ID. This mechanism allows the system to reduce the communication signals that are transmitted in the network since only the tags with the same group ID will respond to the reader's request. Moreover, the grouping mechanism reduces the computation overhead on the reader side when identifying a tag from a large number of tags. As a result, the protocol is scalable by maintaining a consistent operation overhead on both sides of the reader and the tag.

Table 5.1 demonstrates the comparison of the system requirements that are satisfied in our *SLEC* protocol, SAP protocol proposed by Mtita et al. [39], and STS protocol proposed by Sundaresan et al. [5].

5.2 Analysis of Security Requirements

The protocol is based on the Diffie-Hellman digital signature algorithm using a 256-bit key, which is equivalent to the RSA algorithm with a 3072-bit key that is longer than the commonly used key of 2048 [55]. This gives a higher level of security to *SLEC*

algorithm. Therefore, the protocol is secure against different security attacks that most of the RFID protocols can experience.

Table 5.1. Comparison of the System Requirements

System Requirement	SAP	STS	SLEC
Mutual Authentication	Y	Y	Y
Privacy and Confidentiality	N	Y	Y
Message Integrity	Y	Y	Y
Forward and Backward Secrecy	N	N	Y
Anonymity	*	Y	Y
Availability	N	N	Y
Scalability	N	N	Y
Y: satisfied N: not satisfied		*: Not applicable	

1) Replay Attach Resistance

The proposed *SLEC* protocol is secure against replay attack since the authentication session involves timestamps and freshly generated random values as private keys for both reader and tag. If an adversary eavesdrops on the communication channel to replay the tag response, he will not be able to extract any message from the reader or the tag, and the timestamp will not match the current session.

Lemma: SLEC is secure against replay attack

Proof:

Adversary replays old session1 to the reader:

$$M_3 = t_{prv1} \times (r_{prv1} \times P)$$

$$M_4 = h(X_1 + (r_{prv1} \times P) + (t_{prv1} \times P) + Gk_1)$$

Reader verifies in session2:

$$T'_{pub2} = r_{prv2}^{-1} \times [t_{prv1} \times (r_{prv1} \times P)]$$

$$M'_4 = h(X_2 + (r_{prv2} \times P) + [r_{prv2}^{-1} \times t_{prv1} \times (r_{prv1} \times P)] + Gk_2)$$

Verification fails since $M'_4 \neq M_4$. Unauthorized tag is not authenticated.

2) Man-In-The-Middle Attack Resistance

If an adversary interrupts the message transmitted by a reader or a tag, modifies it and sends it back as a real message, the message will not be extracted by any entity. Therefore, the communication will be terminated if no response is sent because all the messages transmitted in the authentication session involve validating the values before extracting any data from them. Therefore, *SLEC* protocol is resistant to MIM attack.

Lemma: SLEC is secure against modification

Proof:

Adversary A intercept message 3 and 4 and modifies the tag information by A:

$$M_3 = a_{prv1} \times (r_{prv1} \times P)$$

$$M_4 = h(X_a + (r_{prv1} \times P) + (a_{prv1} \times P) + Gk_a)$$

Reader verifies in session2:

$$A'_{pub1} = r_{prv1}^{-1} \times [a_{prv1} \times (r_{prv1} \times P)]$$

$$M'_4 = h(X_1 + (r_{prv1} \times P) + [r_{prv1}^{-1} \times t_{prv1} \times (r_{prv1} \times P)] + Gk_1)$$

Verification fails since $M'_4 \neq M_4$ because the tag ID and group key used in the message sent by A are not the same in the reader list for the requested tag. Unauthorized tag is not authenticated.

3) Traceability Attack Resistance

An adversary can trace the signals sent by a specific tag to identify the tag location. However, the reader in *SLEC* protocol broadcasts the message signals to a group of tags that respond to the reader for the same message request. This results in sending different signals from different locations to confuse the adversary from tracking a certain tag to obtain its location. Accordingly, the protocol is resistant to tracing.

Lemma: SLEC is secure against the tracing attack

Proof:

To distinguish the difference between two tags T_1 and T_2, an adversary has to construct the correct hash value with correct tag id (X), timestamp (Ts), and a group key (Gk) which are only transmitted during the setup phase:

$$M_4 = Tag_i + h[(t_{prv1} \times P) + (t_{prv1} \times (y \times P))]$$

$$Tag_i = M_4 - h[(t_{prv1} \times P) + (t_{prv1} \times (y \times P))]$$

The adversary has to solve the correct elliptic curve discrete logarithm problem (ECDLP) to obtain the secret values in Tag_i that are used in the communication.

4) Impersonate Attack Resistance

It is unlikely for any adversary to impersonate the reader or the tag in our protocol since they used a shared point generator algorithm P that is only known to the legitimate server, reader, and tag. So, it is impossible for the adversary to compute the required messages to pass the authentication.

Lemma: SLEC is secure against impersonation

Proof:

Reader: $M_1 = h(Gk)$

$M_2 = (r_{prv} \times P) \oplus Ts$

Adversary: a_{prv}, $A_{pub} = a_{prv} \times P$

$M_{3a} = a_{prv} \times (ra_{prv} \times P)$

$M_{4a} = h(X_a + (ra_{prv} \times P) + A_{pub} + Gk_a)$

Reader: $A'_{pub} = r_{prv}^{-1} \times [a_{prv} \times ra_{prv} \times P)]$

$M_4 = h(X + R'_{pub} + A_{pub} + Gk)$

Validation fails since $M_4 \neq M_{4a}$

Therefore, the unauthorized tag is not authenticated.

5) *Desynchronization Attack Resistance*

The tag in *SLEC* protocol stores the new and previous values of the group identifier that is used at the beginning of the authentication phase. This allows the tag to authenticate the reader if the previous session was interrupted by an adversary to break the synchronization. The communication values are also updated after every successful authentication session using the same algorithm and inputs to maintain the synchronization state between the network entities.

6) Denial of Service Attack Resistance

In our *SLEC* protocol, the reader and the tag generate their keys separately using the same key generation algorithm, so there is no synchronous update of the keys between the server and the tag for the attack to occur.

Table 5.2 demonstrates the comparison of the security attacks resistance between our *SLEC* protocol, SAP protocol proposed by Mtita et al. [39], and STS protocol proposed by Sundaresan et al. [5].

Table 5.2 Comparison of the Security Threats Resistance

Attacks	**SAP**	**STS**	**SLEC**
Replay Attack	Y	Y	Y
Man-in-the-Middle	*	*	Y
Eavesdropping	Y	*	Y
Impersonate Attack	Y	Y	Y
Traceability Attack	Y	Y	Y
Desynchronization	*	Y	Y
Denial of Service	*	Y	Y
Y: Satisfied	N: Not satisfied		*: Not applicable

5.3 Analysis of Computation Cost

Since the passive tag used in the RFID system has limited capabilities and resources, it is essential to consider the computation and security features for the appropriate application. Even though the elliptic curve has higher computation overhead on both the reader and the tag, we provide a higher security level in our *SLEC* protocol that satisfies the resistance to all the security attacks. Moreover, we compare our protocol with additional server-based elliptic curve protocols such as IECC protocol proposed by Farash [15] and EECC protocol proposed by Zhang and Qi [16] to illustrate a well-defined measurement for the computation complexity. The comparison shows that there is no major additional cost between the previously proposed ECC-based protocols and our *SLEC* protocol, although our protocol is completely serverless in the authentication

phase. Table 5.3 demonstrates the operations computed by the tag and the number of transmitted messages from the reader and the tag during the authentication phase.

Table 5.3 Comparison of the Computation Cost on the Tag

Protocol	**Operation**	**Tag**	**Reader**
SAP [39]	$2T_H + 2T_{RNG}$	1	2
STS [5]	$7T_{XOR} + 3T_{PRNG}$	1	1
IECC [15]	$2T_{SMUL} + 2T_H$	1	2
EECC [16]	$2T_{SMUL} + T_{SAD} + 2T_H$	1	2
SLEC	$2T_{SMUL} + 3T_H$	1	2

T_{SMUL}: scalar multiplication, T_{SAD}: scalar addition, T_H: one-way hash, T_{XOR}: XOR, T_{PRNG}: pseudo-random number generation

.

CHAPTER 6: FORMAL VERIFICATION

The formal verification of the protocol is done to prove the correctness of the algorithms used in the protocol in terms of security and authentication. ProVerif tool [56] is one of the powerful tools to analyze the security of cryptographic protocols. It is an automatic cryptographic protocol verifier that is developed by Bruno Blanchet to validate the security and authentication properties of the cryptographic algorithms in formal models.

In this section, we use the ProVerif tool to validate reachability and secrecy (security), and correspondence assertion (authentication) of SLEC protocol. The results of the verification process are also presented.

6.1 Adversary Model

The ProVerif tool is based on a model where the adversary can intercept, alter, and inject the messages into an insecure network. In SLEC protocol, the adversary has initial knowledge of the finite set of parameters that increase during the protocol execution in parallel with the adversary. No matter how the adversary interacts with the protocol, ProVerif verifies the secrecy of the messages and values transmitted between the server, the reader, and the tag. Therefore, the secret messages will never be a part of the adversary knowledge to run the protocol successfully. The results of the ProVerif verification in this section show that the protocol preserves the secrecy of the messages and values that intercepting or altering the message will lead to the protocol termination.

6.2 Reachability and Secrecy

Reachability and secrecy in ProVerif analyze the security properties of the protocol against any attacker. We investigate the reachability of a term *x* by an adversary *A*, so we assess the secrecy of *x* concerning the modeled protocol. In SLEC protocol, we use ProVerif to test whether the secret messages in the setup phase "Ms", and the secret messages in the authentication phase "Ma" are not available to an adversary *A*. We represent the messages transmitted in the setup phase from the server, the reader, and the tag as "Mss", "Msr" and "Mst" respectively. Moreover, we represent the messages transmitted in the authentication phase from the reader and the tag as "Mar" and "Mat" respectively. The complete verification process is demonstrated in Figure 6.2. The results of the verification process conclude, "RESULT, not attacker(Mst1[]) is true" which means the setup phase message M_1 from the tag is unreachable, and an attack cannot be conducted against the protocol successfully. Similarly, "RESULT not attacker(Mar1[]) is true" means the authentication phase message M_1 from the reader is unreachable and secure against the attacks. All setup phase and authentication phase messages are tested and resulted in true reachability and secrecy proof.

Reachability and Secrecy
Process: {1}new p: P; {2}new y_22: Y; {3}new tprv1: pu_pr_key;

```
{4}new rprv1: pu_pr_key;
{5}new xi: TagID;
{6}new Gxi: GK;
(
    {7}!
    {8}let Tpub1: pu_pr_key = find_R_and_Tpubs(p,tprv1) in
    {9}let Mst1_23: pu_pr_key = setPhase_encrypt(tprv1,y_22) in
    {10}out(ch, Mst1_23);
    {11}in(ch, Mss4': Tag_and_Hash);
    {12}let tag': Tag = verify_tag(Mss4',h(Tpub1,Mst1_23)) in
    {13}in(ch, ms1: xored_key);
    {14}let Ms1': GK = validate_Ms1(ms1) in
    {15}new ggkk: GK;
    {16}if (Ms1' = ggkk) then
    {17}new tprv2: pu_pr_key;
    {18}let Tpub2: pu_pr_key = find_R_and_Tpubs(p,tprv2) in
    {19}in(ch, ms2: pu_pr_key);
    {20}new ts2': timeStampes;
    {21}let Rpub2': pu_pr_key = xor(ms2,ts2') in
    [29]let Mat3_24: pu_pr_key = authen_phase_encrypt(tprv2,Rpub2') in
    {23}out(ch, Mat3_24);
    {24}new xi_25: TagID;
```

```
    {25}let Mat4_26: Hashing = h_ph2_2(xi_25,Rpub2',Tpub1,Ms1') in

    {26}out(ch, Mat4_26);

    {27}let xxi: TagID = PRNG(xi_25) in

    {28}let GKi: GK = PRNG_Group(ggkk) in

    {29}in(ch, Mar5': Hashing);

    {30}let Mar55: Hashing = h_ph2_3(xi_25,Tpub2,Rpub2') in

    {31}if (Mar55 = Mar5') then

    0

) | (

    {32}!

    {33}let Rpub1: pu_pr_key = find_R_and_Tpubs(p,rprv1) in

    {34}let Msr2_27: pu_pr_key = setPhase_encrypt(rprv1,y_22) in

    {35}out(ch, Msr2_27);

    {36}in(ch, Mss3': TagList_and_Hash);

    {37}let taglisht': TageList = verify_tagList(Mss3',h(Rpub1,Msr2_27)) in

    {38}new rprv2: pu_pr_key;

    {39}let Rpub2: pu_pr_key = find_R_and_Tpubs(p,rprv2) in

    {40}new GKii: GK;

    {41}let Mar1_28: hash_GK = h_ph2_(GKii) in

    {42}out(ch, Mar1_28);

    {43}new ts2: timeStampes;

    {44}let Mar2_29: pu_pr_key = xor(Rpub2,ts2) in
```

```
    {45}out(ch, Mar2_29);
    {46}in(ch, Mat3': pu_pr_key);
    {47}let Tpub2': pu_pr_key = authen_phase_dencrypt(Mat3',re_pu_pr_key(rprv2)) in
    {48}in(ch, Mat4': Hashing);
    {49}new xii': TagID;
    {50}let Mar5: Hashing = h_ph2_3(xii',Rpub2,Tpub2') in
    {51}out(ch, Mar5)
) | (
    {52}!
    {53}in(ch, (Mst1': pu_pr_key,Msr1': pu_pr_key));
    {54}let Tprv1': pu_pr_key = setPhase_decrypt(Mst1',getYinv(y_22)) in
    {55}let Rprv1': pu_pr_key = setPhase_decrypt(Msr1',getYinv(y_22)) in
    {56}new ts: timeStampes;
    {57}let tagi: Tag = create_tag(xi,ts,Gxi) in
    {58}let tagListi: TageList = create_taglist(tagi) in
    {59}let Mss3_30: TagList_and_Hash = prepare_tagList(tagListi,h(Rprv1',Msr1')) in
    {60}out(ch, Mss3_30);
    {61}let Mss4_31: Tag_and_Hash = prepare_tag(tagi,h(Tprv1',Mst1')) in
    {62}out(ch, Mss4_31)
)
-- Query not attacker(Mst1[])
Completing...
```

Starting query not attacker(Mst1[])

RESULT not attacker(Mst1[]) is true.

-- Query not attacker(Msr2[])

Completing...

Starting query not attacker(Msr2[])

RESULT not attacker(Msr2[]) is true.

-- Query not attacker(Mss3[])

Completing...

Starting query not attacker(Mss3[])

RESULT not attacker(Mss3[]) is true.

-- Query not attacker(Mss4[])

Completing...

Starting query not attacker(Mss4[])

RESULT not attacker(Mss4[]) is true.

-- Query not attacker(Mar1[])

Completing...

Starting query not attacker(Mar1[])

RESULT not attacker(Mar1[]) is true.

-- Query not attacker(Mar2[])

Completing...

Starting query not attacker(Mar2[])

RESULT not attacker(Mar2[]) is true.

-- Query not attacker(Mat3[])

Completing...

Starting query not attacker(Mat3[])

RESULT not attacker(Mat3[]) is true.

-- Query not attacker(Mat4[])

Completing...

Starting query not attacker(Mat4[])

RESULT not attacker(Mat4[]) is true.

Figure 6.2: Verification results of reachability and secrecy

6.3 Correspondence Assertion

The correspondence assertion in ProVerif is to model the authentication of the protocol using a sequence of events. We apply a sequence of events to verify the authentication of the reader to the tag and the authentication of the tag to the reader through the encrypted messages individually. The complete verification process of authentication is presented in Figure 6.2. The results of the correspondence assertion verification show "RESULT inj-event(termReader(x)) ==> inj-event(acceptsReader(x)) is true" which means the reader is authenticated by the tag, and "RESULT inj-event(termTag(x_24)) ==> inj-event(acceptsTag(x_24)) is true" which means the tag is authenticated by the reader. The verification results confirm that SLEC protocol achieves a successful mutual authentication between the reader and the tag.

Correspondence Assertion

```
Process:
{1}new p: P;
{2}new y_22: Y;
{3}new tprv1: pu_pr_key;
{4}new rprv1: pu_pr_key;
{5}new xi: TagID;
{6}new Gxi: GK;
(
    {7}!
    {8}let Tpub1: pu_pr_key = find_R_and_Tpubs(p,tprv1) in
    {9}let Mst1: pu_pr_key = setPhase_encrypt(tprv1,y_22) in
    {10}out(ch, Mst1);
    {11}in(ch, Mss4': Tag_and_Hash);
    {12}let tag': Tag = verify_tag(Mss4',h(Tpub1,Mst1)) in
    {13}in(ch, ms1: xored_key);
    {14}let Ms1': GK = validate_Ms1(ms1) in
    {15}new ggkk: GK;
    {16}if (Ms1' = ggkk) then
    {17}new tprv2: pu_pr_key;
    {18}let Tpub2: pu_pr_key = find_R_and_Tpubs(p,tprv2) in
    {19}in(ch, ms2: pu_pr_key);
```

```
    {20}new ts2': timeStampes;
    {21}let Rpub2': pu_pr_key = xor(ms2,ts2') in
    {22}let Mat3: pu_pr_key = authen_phase_encrypt(tprv2,Rpub2') in
    {23}out(ch, Mat3);
    {24}new xi_23: TagID;
    [29]let Mat4: Hashing = h_ph2_2(xi_23,Rpub2',Tpub1,Ms1') in
    {26}out(ch, Mat4);
    {27}let xxi: TagID = PRNG(xi_23) in
    {28}let GKi: GK = PRNG_Group(ggkk) in
    {29}in(ch, Mar5': Hashing);
    {30}let Mar55: Hashing = h_ph2_3(xi_23,Tpub2,Rpub2') in
    {31}if (Mar55 = Mar5') then
    {32}event acceptsReader(Mar55);
    {33}event termReader(Mar55)
) | (
    {34}!
    {35}let Rpub1: pu_pr_key = find_R_and_Tpubs(p,rprv1) in
    {36}let Msr2: pu_pr_key = setPhase_encrypt(rprv1,y_22) in
    {37}out(ch, Msr2);
    {38}in(ch, Mss3': TagList_and_Hash);
    {39}let taglisht': TageList = verify_tagList(Mss3',h(Rpub1,Msr2)) in
    {40}new rprv2: pu_pr_key;
```

```
    {41}let Rpub2: pu_pr_key = find_R_and_Tpubs(p,rprv2) in
    {42}new GKii: GK;
    {43}let Mar1: hash_GK = h_ph2_(GKii) in
    {44}out(ch, Mar1);
    {45}new ts2: timeStampes;
    {46}let Mar2: pu_pr_key = xor(Rpub2,ts2) in
    {47}out(ch, Mar2);
    {48}in(ch, Mat3': pu_pr_key);
    {49}let Tpub2': pu_pr_key = authen_phase_dencrypt(Mat3',re_pu_pr_key(rprv2)) in
    {50}in(ch, Mat4': Hashing);
    {51}new xii': TagID;
    {52}let Mar5: Hashing = h_ph2_3(xii',Rpub2,Tpub2') in
    {53}event acceptsTag(Mar5);
    {54}out(ch, Mar5);
    {55}event termTag(Mar5)
) | (
    {56}!
    {57}in(ch, (Mst1': pu_pr_key,Msr1': pu_pr_key));
    {58}let Tprv1': pu_pr_key = setPhase_decrypt(Mst1',getYinv(y_22)) in
    {59}let Rprv1': pu_pr_key = setPhase_decrypt(Msr1',getYinv(y_22)) in
    {60}new ts: timeStampes;
    {61}let tagi: Tag = create_tag(xi,ts,Gxi) in
```

```
    {62}let tagListi: TageList = create_taglist(tagi) in
    {63}let Mss3: TagList_and_Hash = prepare_tagList(tagListi,h(Rprv1',Msr1')) in
    {64}out(ch, Mss3);
    {65}let Mss4: Tag_and_Hash = prepare_tag(tagi,h(Tprv1',Mst1')) in
    {66}out(ch, Mss4)
)
-- Query inj-event(termReader(x)) ==> inj-event(acceptsReader(x))
Completing...
Starting query inj-event(termReader(x)) ==> inj-event(acceptsReader(x))
RESULT inj-event(termReader(x)) ==> inj-event(acceptsReader(x)) is true.
-- Query inj-event(termTag(x_24)) ==> inj-event(acceptsTag(x_24))
Completing...
Starting query inj-event(termTag(x_24)) ==> inj-event(acceptsTag(x_24))
RESULT inj-event(termTag(x_24)) ==> inj-event(acceptsTag(x_24)) is true.
```

Figure 6.3: Verification results of correspondence assertions

CHAPTER 7: CONCLUSION

RFID is the new alternative to physical barcoding, which is being widely used in the fields of product authentication and database storage. Serverless RFID protocols are being developed to provide a dynamic network that the mobile tags can be searched and identified in different locations away from the server. As RFID network carries along with sensitive information, and passive tags have limited resources, many security algorithms have been deduced and implemented at a minimal cost using simple operations that are still vulnerable to security attacks. We propose a secure serverless RFID protocol (*SLEC*) that uses the elliptic curve cryptography based on the Diffie-Hellman algorithm. The algorithm used is classified as a public key algorithm that can be handled by low constraint devices such as RFID passive tags. *SLEC* protocol is considered to be secure against different security attacks that simple protocols suffer from. The reader in *SLEC* protocol is completely capable of identifying and authenticating mobile tags under an offline server. We also introduced the tag grouping mechanism to reduce the computation overhead on the reader side that is elevated from the scalar operations in the elliptic curve computation when identifying a tag in a large-scale network, and also to create a scalable system that is not affected by the tag population size. Also, *SLEC* protocol has a recovery mechanism that any compromised values can soon be renewed by any server in the network to retrieve the tag and protocol privacy. The protocol is tested using ProVerif cryptographic verification tool to prove that SLEC achieves successful security and authentication. We believe that the widespread implementation of serverless RFID systems will improve the efficiency of all

businesses and processes. This will take RFID technology into the upper level by driving the world to go smart.

REFERENCES

[1] L. Xie, Y. Yin, A. V. Vasilakos, and S. Lu, "Managing RFID Data: Challenges, Opportunities and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1294-1311, 2014.

[2] J. Pagán Alexander, R. Baashirah, and A. Abuzneid, "Comparison and Feasibility of Various RFID Authentication Methods Using ECC," *Sensors*, vol. 18, no. 9, p. 2902, 2018.

[3] EPCglobal. Inc., " EPCTM Radio-Frequency Identity Protocols Generation-2 UHF RFID," in *Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz Version 2.0.1 Ratified*, ed. Lawrenceville, NJ, USA: EPCglobal Inc., 2015.

[4] R. Baashirah and A. Abuzneid, "Survey on Prominent RFID Authentication Protocols for Passive Tags," *Sensors*, vol. 18, no. 10, 2018.

[5] S. Sundaresan, R. Doss, S. Piramuthu, and W. Zhou, "Secure Tag Search in RFID Systems Using Mobile Readers," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 2, pp. 230-242, 2015.

[6] J. S. Chou, "An efficient mutual authentication RFID scheme based on elliptic curve cryptography," *The Journal of Supercomputing*, vol. 70, no. 1, pp. 75-94, 2014.

[7] L. Zheng, Y. Xue, L. Zhang, and R. Zhang, "Mutual Authentication Protocol for RFID Based on ECC," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 2, pp. 320-323, 2017.

[8] M. R. Alagheband and M. R. Aref, "Simulation-Based Traceability Analysis of RFID Authentication Protocols," *Wireless Personal Communications*, vol. 77, no. 2, pp. 1019-1038, 2014.

[9] J. Wang, C. Floerkemeier, and S. E. Sarma, "Session-based security enhancement of RFID systems for emerging open-loop applications," *Personal and Ubiquitous Computing*, vol. 18, no. 8, pp. 1881-1891, 2014.

[10] International Organization for Standardization, "ISO/IEC DIS 9798-2," in *Information technology - Security techniques - Entity authentication - Part 2: Mechanisms using authenticated encryption*, ed. Switzerland: International Organization for Standardization, 2017.

[11] E. K. Ryu, D. S. Kim, and K. Y. Yoo, "On Elliptic Curve Based Untraceable RFID Authentication Protocols," presented at the Proceedings of the 3rd ACM

Workshop on Information Hiding and Multimedia Security, Portland, Oregon, USA, 2015.

[12] R. Songhela and M. L. Das, "Yet Another Strong Privacy-Preserving RFID Mutual Authentication Protocol," Cham, 2014, pp. 171-182: Springer International Publishing.

[13] Y. Chen and J. S. Chou, "ECC-based untraceable authentication for large-scale active-tag RFID systems," *Electronic Commerce Research,* journal article vol. 15, no. 1, pp. 97-120, 2015.

[14] Q. Yao, J. Ma, S. Cong, X. Li, and J. Li, "Attack gives me power: DoS-defending constant-time privacy-preserving authentication of low-cost devices such as backscattering RFID tags," presented at the Proceedings of the 3rd ACM Workshop on Mobile Sensing, Computing and Communication, Paderborn, Germany, 2016.

[15] M. S. Farash, "Cryptanalysis and improvement of an efficient mutual authentication RFID scheme based on elliptic curve cryptography," *The Journal of Supercomputing,* vol. 70, no. 2, pp. 987-1001, 2014.

[16] Z. Zhang and Q. Qi, "An Efficient RFID Authentication Protocol to Enhance Patient Medication Safety Using Elliptic Curve Cryptography," *Journal of Medical Systems,* vol. 38, no. 5, p. 47, 2014.

[17] R. Baashirah, A. Abuzneid, A. Tammineedi, M. Mathew, N. Bandaru, and P. Delair, "Improve Healthcare Safety Using Hash-Based Authentication Protocol for RFID Systems," presented at the The 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON, New York, NY, 2018.

[18] B. C. Chen, C. T. Yang, H. T. Yeh, and C. C. Lin, "Mutual Authentication Protocol for Role-Based Access Control Using Mobile RFID," *Applied Sciences*, vol. 6, no. 8, p. 215, 2016.

[19] H. Fernando and J. Abawajy, "Mutual Authentication Protocol for Networked RFID Systems," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 417-424, 2011.

[20] X. Chen, T. Cao, and J. Zhai, "Untraceability Analysis of Two RFID Authentication Protocols," *Chinese Journal of Electronics*, vol. 25, no. 5, pp. 912-920, 2016.

[21] C. Chen, Z. Qian, I. You, J. Hong, and S. Lu, "ACSP: A Novel Security Protocol against Counting Attack for UHF RFID Systems," in *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 100-105, 2011.

[22] M. Safkhani, N. Bagheri, and A. Mahani, "On the security of RFID anti-counting security protocol (ACSP)," *Journal of Computational and Applied Mathematics,* vol. 259, pp. 512-521, 2014.

[23] H.-Y. Chien and C.-W. Huang, "A Lightweight Authentication Protocol for Low-Cost RFID," *Journal of Signal Processing Systems*, journal article vol. 59, no. 1, pp. 95-102, 2010.

[24] Y. z. Li, Y. b. Cho, N. k. Um, and S. h. Lee, "Security and Privacy on Authentication Protocol for Low-cost RFID," in *2006 International Conference on Computational Intelligence and Security*, vol. 2, pp. 1101-1104, 2006.

[25] International Organization for Standardization, "ISO/IEC 15693-2:2006," in *Identification cards - Contactless integrated circuit cards - Vicinity cards - Part 2: Air interface and initialization*, ed. Switzerland: International Organization for Standardization, 2006.

[26] M. Burmester and J. Munilla, "Lightweight RFID authentication with forward and backward security," *ACM Trans. Inf. Syst. Secur.,* vol. 14, no. 1, pp. 1-26, 2011.

[27] S. Lee, T. Asano, and K. Kim, "RFID Mutual Authentication Scheme based on Synchronized Secret Information," in *Symposium on Cryptography and Information Security*, Hiroshima, Japan, 2006.

[28] Y. Zuo, "Survivability Experiment and Attack Characterization for RFID," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 289-302, 2012.

[29] K. Lee, J. M. Gonz, #225, l. Nieto, and C. Boyd, "Improving the efficiency of RFID authentication with pre-computation," presented at the Proceedings of the Tenth Australasian Information Security Conference - Volume 125, Melbourne, Australia, 2012.

[30] F. Rahman and S. I. Ahamed, "DRAP: a Robust Authentication protocol to ensure survivability of computational RFID networks," presented at the Proceedings of the 27th Annual ACM Symposium on Applied Computing, Trento, Italy, 2012.

[31] H. Liu, H. Ning, Y. Zhang, D. He, Q. Xiong, and L. T. Yang, "Grouping-Proofs-Based Authentication Protocol for Distributed RFID Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1321-1330, 2013.

[32] F. Rahman and S. I. Ahamed, "Efficient detection of counterfeit products in large-scale RFID systems using batch authentication protocols," *Personal and Ubiquitous Computing*, journal article vol. 18, no. 1, pp. 177-188, 2014.

[33] Y. Keqiang, S. Lingling, Q. Xing, and Z. Zhonghua, "Design of anti-collision integrated security mechanism based on chaotic sequence in UHF RFID system," *China Communications*, vol. 11, no. 3, pp. 137-147, 2014.

[34] J.-S. Cho, Y.-S. Jeong, and S. O. Park, "Consideration on the brute-force attack cost and retrieval cost: A hash-based radio-frequency identification (RFID) tag mutual authentication protocol," *Computers & Mathematics with Applications,* vol. 69, no. 1, pp. 58-65, 2015.

[35] C. C. Chang, W. Y. Chen, and T. F. Cheng, "A Secure RFID Mutual Authentication Protocol Conforming to EPC Class 1 Generation 2 Standard," in *2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2014, pp. 642-645.

[36] Z. Liu, D. Liu, L. Li, H. Lin, and Z. Yong, "Implementation of a New RFID Authentication Protocol for EPC Gen2 Standard," *IEEE Sensors Journal,* vol. 15, no. 2, pp. 1003-1011, 2015.

[37] H. Niu, E. Taqieddin, and S. Jagannathan, "EPC Gen2v2 RFID Standard Authentication and Ownership Management Protocol," *IEEE Transactions on Mobile Computing,* vol. 15, no. 1, pp. 137-149, 2016.

[38] P. Dass and H. Om, "A Secure Authentication Scheme for RFID Systems," *Procedia Computer Science,* vol. 78, pp. 100-106, 2016.

[39] C. Mtita, M. Laurent, and J. Delort, "Efficient serverless radio-frequency identification mutual authentication and secure tag search protocols with untrusted readers," *IET Information Security,* vol. 10, no. 5, pp. 262-271, 2016.

[40] B. Blanchet, "Composition Theorems for CryptoVerif and Application to TLS 1.3," presented at the IEEE 31st Computer Security Foundations Symposium (CSF), Oxford, UK, 2010. Available: http://prosecco.gforge.inria.fr/personal/bblanche/cryptoverif/

[41] R. Aggarwal and M. L. Das, "RFID security in the context of "internet of things"," presented at the Proceedings of the First International Conference on Security of Internet of Things, Kollam, India, 2012.

[42] Y. C. Chen, W. L. Wang, and M. S. Hwang, "RFID Authentication Protocol for Anti-Counterfeiting and Privacy Protection," in *The 9th International Conference on Advanced Communication Technology*, vol. 1, pp. 255-259, 2007.

[43] Y. J. Huang, W. C. Lin, and H. L. Li, "Efficient Implementation of RFID Mutual Authentication Protocol," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 12, pp. 4784-4791, 2012.

[44] International Organization for Standardization, "ISO/IEC 18000-6:2013," in *Information technology - Radio frequency identification for item management - Part 6: Parameters for air interface communications at 860 MHz to 960 MHz General*, ed. Switzerland: International Organization for Standardization, 2013.

[45] Y. C. Huang and J. R. Jiang, "Ultralightweight RFID Reader-Tag Mutual Authentication," in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 3, pp. 613-616, 2015.

[46] H.-Y. Chien and C.-H. Chen, "Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards," *Computer Standards & Interfaces,* vol. 29, no. 2, pp. 254-259, 2007.

[47] C.-L. Chen and Y.-Y. Deng, "Conformation of EPC Class 1 Generation 2 standards RFID system with mutual authentication and privacy protection," *Engineering Applications of Artificial Intelligence,* vol. 22, no. 8, pp. 1284-1291, 2009.

[48] N. J. Hopper and M. Blum, "Secure Human Identification Protocols," presented at the Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, 2001.

[49] S. Piramuthu, "HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication," in *CollECTeR Europe Conference,* 2006.

[50] Z. Lin and J. S. Song, "An Improvement in HB-Family Lightweight Authentication Protocols for Practical Use of RFID System," *Journal of Advances in Computer Networks,* vol. 1, no. 1, 2013.

[51] A. Juels and S. A. Weis, "Authenticating Pervasive Devices with Human Protocols," Berlin, Heidelberg, 2005, pp. 293-308: Springer Berlin Heidelberg.

[52] H. Gilbert, M. Robshaw, and H. Sibert, "Active attack against HB/sup +/: a provably secure lightweight authentication protocol," *Electronics Letters*, vol. 41, no. 21, pp. 1169-1170, 2005.

[53] M. Ouaskou, M. Lahmer, and M. Belkasmi, "A variant of HB protocols based on permutation for low-cost RFID," in *2015 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1-4, 2015.

[54] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976.

[55] A. Kumar *et al.*, "Elliptic Curve Cryptography (ECC) Certificates Performance Analysis," Symantec, United States2013, Available: https://www.websecurity.symantec.com/content/dam/websitesecurity/digitalassets/desktop/pdfs/whitepaper/Elliptic_Curve_Cryptography_ECC_WP_en_us.pdf.

[56] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre. ProVerif: Automatic Cryptographic Protocol Verier, User Manual and Tutorial [Online]. Available: https://prosecco.gforge.inria.fr/personal/bblanche/proverif/manual.pdf